



UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

CORSO DI LAUREA IN INFORMATICA E

COMUNICAZIONE DIGITALE

Sede: Taranto

ADVENTURE WITH SIMUNAV

Analisi, progettazione e realizzazione delle modifiche del progetto del gioco di avventura

“L'astronave condannata”, presentato durante il corso di Algoritmi e Strutture Dati

DOCENTE: Stefano Ferilli

1	TRACCIA	11
1.1	MODIFICA ANTONIO PASTORELLI	11
1.2	MODIFICA ANTONIO BASILE.....	11
1.3	MODIFICA DEL GIUDICE ANGELO	11
1.4	MODIFICA MOSCHETTI MARCO	11
1.5	MODIFICA GERMANO GALEANDRO.....	11
1.6	MODIFICA D'ANDRIA IVAN – DRESDA CLAUDIO	12
1.7	MODIFICA FEDERICA FORTE	12
1.8	MODIFICA GIANLUCA VACCA	12
1.9	MODIFICA SALVATORE VESTITA	12
1.10	MODIFICA VINCENZO GIANNUZZO	12
1.11	MODIFICA CICALA GIACOMO	12
1.12	MODIFICA SCATIGNA GIANLUCA.....	12
1.13	MODIFICA DELLA FOLGORÉ GRAZIA.....	12
1.14	MODIFICA PALAGIANO MARCELLO.....	12
1.15	MODIFICHE CHIARAPPA ROSA.....	13
1.16	MODIFICHE LUCERI MATTEO	13
2	ANALISI.....	13
2.1	MUSEO	13
2.2	SCUOLA E AULE	13
2.3	ARCHIVIO E MUSEO (VECCIA VERSIONE DEL MUSEO, INSERITA PER COMPLETEZZA).....	14
2.4	AULA SINGOLA.....	14
2.5	UFFICIO POSTALE	14
2.6	DIALOGHI.....	15
2.7	SALA SCOMMESSE	15
2.8	SIMULATORE.....	15
2.9	AUDITORIUM	15
2.10	STAZIONE	15
2.11	BIBLIOTECA	16
2.12	BARI – ROMA – PISA	16
2.13	MECCANICO	16
2.14	VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”	17
2.15	ZAINO E VALIGIA	17
3	PROGETTAZIONE	18

3.1	MUSEO	18
3.2	SCUOLA E AULE	18
3.2.1	<i>Luogo Scuola</i>	18
3.2.2	<i>Luogo Classe 1A</i>	18
3.2.3	<i>Luogo Classe 2A</i>	18
3.2.4	<i>Luogo Classe 3A</i>	19
3.3	ARCHIVIO E MUSEO (VECCHIA VERSIONE DEL MUSEO).....	19
3.3.1	<i>Luogo Archivio</i>	19
3.3.2	<i>File</i>	19
3.3.3	<i>Museo</i>	19
3.4	AULA SINGOLA.....	20
3.4.1	<i>Computer</i>	20
3.4.2	<i>Sedie</i>	20
3.4.3	<i>Proiettore</i>	20
3.4.4	<i>Libri</i>	20
3.4.5	<i>Lezione</i>	20
3.5	UFFICIO POSTALE	21
3.6	DIALOGHI.....	21
3.7	SALA SCOMMESSE E SIMULATORE	22
3.7.1	<i>Azioni</i>	22
3.7.2	<i>Inserimento Azione "Guarda simulatore"</i>	22
3.7.3	<i>Inserimento Azione "Avvia simulatore" e "Gioca simulatore"</i>	22
3.8	AUDITORIUM	23
3.9	STAZIONE.....	23
3.10	PALESTRA	24
3.11	SALA GIOCHI.....	24
3.11.1	<i>Slot machine</i>	25
3.12	BIBLIOTECA	25
3.13	BARI – ROMA – PISA	26
3.14	MECCANICO.....	26
3.15	CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”	27
3.16	ZAINO E VALIGIA	27
4	REALIZZAZIONE	29
4.1	MUSEO	29

4.2	SCUOLA E AULE	30
4.2.1	<i>Scuola e le classi 1A, 2A, 3A</i>	30
4.2.2	<i>Oggetti.....</i>	30
4.2.3	<i>Azioni.....</i>	31
4.3	ARCHIVIO E MUSEO (VECCHIA VERSIONE DEL MUSEO).....	31
4.3.1	<i>Archivio e Museo</i>	32
4.3.2	<i>Oggetti.....</i>	32
4.3.3	<i>Azioni.....</i>	33
4.4	AULA SINGOLA.....	33
4.4.1	<i>Aula</i>	33
4.4.2	<i>Oggetti.....</i>	33
4.4.3	<i>Azioni.....</i>	34
4.4.4	<i>Comandi.....</i>	34
4.5	UFFICIO POSTALE	35
4.5.1	<i>Mappa.nav.....</i>	35
4.5.2	<i>24.txt</i>	35
4.5.3	<i>Inserimento oggetto "Sportello Telematico"</i>	35
4.5.4	<i>Inserimento oggetto "Documento d'Identità"</i>	36
4.5.5	<i>Inserimento oggetto "Lettera"</i>	37
4.5.6	<i>Inserimento oggetto "Pacco"</i>	37
4.6	DIALOGHI.....	37
4.7	SALA SCOMMESSE E SIMULATORE	38
4.7.1	<i>Aggiunta oggetti e vocaboli nel gioco</i>	38
4.8	AUDITORIUM	38
4.8.1	<i>Classi create</i>	38
4.8.2	<i>Realizzazioni usate</i>	39
4.8.3	<i>Modifica Mappa.....</i>	39
4.8.4	<i>Oggetti.....</i>	40
4.8.5	<i>Azioni.....</i>	40
4.9	STAZIONE.....	41
4.10	PALESTRA	42
4.10.1	<i>Oggetti</i>	42
4.10.2	<i>Azioni.....</i>	43
4.10.3	<i>Comandi</i>	43

4.11	SALA GIOCHI	44
4.11.1	<i>Oggetto</i>	44
4.11.2	<i>Azioni</i>	45
4.11.3	<i>Comandi</i>	45
4.12	BIBLIOTECA	45
4.12.1	<i>Oggetti</i>	45
4.12.2	<i>Azioni</i>	46
4.12.3	<i>Comandi</i>	46
4.13	BARI – ROMA – PISA	46
4.13.1	<i>Oggetti</i>	47
4.13.2	<i>Azioni</i>	47
4.14	MODIFICA CODICI LUOGHI – PALAGIANO MARCELLO	48
4.14.1	<i>Modifica Mappa</i>	48
4.14.2	<i>Modifica MappaOsservatorio</i>	48
4.14.3	<i>Modifica MappaAliena</i>	48
4.15	MECCANICO	49
4.15.1	<i>Oggetti</i>	49
4.15.2	<i>Azioni</i>	49
4.16	CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”	50
4.16.1	<i>Verbi “ruba” e “rubati”</i>	50
4.16.2	<i>Personaggio “carabiniere”</i>	51
4.16.3	<i>Luogo “caserma”</i>	51
4.17	ZAINO E VALIGIA	52
5	IMPLEMENTAZIONE.....	54
5.1	MUSEO	54
5.1.1	<i>Museo.h</i>	54
5.1.2	<i>Museo.cpp</i>	55
5.1.3	<i>Lista.h</i>	69
5.1.4	<i>Cella_L_DP.h</i>	74
5.1.5	<i>Grafo.h</i>	76
5.1.6	<i>Adiacenza.h</i>	84
5.1.7	<i>Modifiche relative ad Astro.cpp</i>	87
5.2	SCUOLA E AULE	91
5.2.1	<i>Astro.h</i>	91

5.2.2	<i>Gioco.h</i>	91
5.2.3	<i>Astro.cpp</i>	91
5.2.4	<i>Mappa.nav</i>	109
5.2.5	<i>Aggiunta dei file nella cartella descrizioni</i>	110
5.3	ARCHIVIO E MUSEO (VECCHIA VERSIONE DEL MUSEO).....	111
5.3.1	<i>Astro.h</i>	111
5.3.2	<i>Gioco.h</i>	111
5.3.3	<i>Astro.cpp</i>	111
5.3.4	<i>Gioco.cpp</i>	125
5.3.5	<i>Mappa.nav</i>	154
5.3.6	<i>Descrizioni</i>	154
5.4	AULA SINGOLA.....	155
5.4.1	<i>AGGIORNAMENTO DEI VOCABOLI DEL DIZIONARIO</i>	155
5.4.2	<i>Aggiornamento del dizionario degli oggetti</i>	155
5.4.3	<i>Aggiornamento delle azioni di gioco</i>	155
5.4.4	<i>Implementazione dell'Aula</i>	166
5.4.5	<i>Correzione Funzionalità usa computer (Andrea Tursi)</i>	166
5.5	UFFICIO POSTALE	168
5.5.1	<i>Inserimento del luogo "Ufficio Postale"</i>	168
5.5.2	<i>Gioco.h</i>	170
5.5.3	<i>Codice per inserimento oggetti</i>	170
5.5.4	<i>Nuove azioni</i>	171
5.5.5	<i>Modifiche al progetto di Bellanova</i>	179
5.6	DIALOGHI.....	180
5.6.1	<i>Implementazione delle variabili</i>	180
5.6.2	<i>Implementazione dei metodi</i>	180
5.6.3	<i>Modifiche dei metodi già presenti in Gioco.cpp</i>	184
5.6.4	<i>File aggiunti</i>	194
5.7	SALA SCOMMESSE E SIMULATORE	209
5.7.1	<i>Implementazione luogo Sala scommesse e nuove azioni nel gioco</i>	209
5.7.2	<i>Strutture dati aggiuntive</i>	213
5.8	AUDITORIUM	218
5.8.1	<i>Gioco.h</i>	218
5.8.2	<i>Astro.h</i>	219

5.8.3	<i>Astro.cpp</i>	219
5.8.4	<i>Gioco.cpp</i>	228
5.9	STAZIONE.....	229
5.9.1	<i>Mappa.nav</i>	230
5.9.2	<i>Descrizioni (33.txt)</i>	230
5.9.3	<i>Astro.cpp</i>	230
5.9.4	<i>Astro.h</i>	237
5.9.5	<i>Biglietto.h</i>	237
5.9.6	<i>Biglietto.cpp</i>	238
5.10	PALESTRA	239
5.10.1	<i>Mappa.nav</i>	239
5.10.2	<i>Descrizioni (34.txt)</i>	239
5.10.3	<i>Astro.cpp</i>	240
5.10.4	<i>Astro.h</i>	242
5.10.5	<i>Gioco.cpp</i>	243
5.10.6	<i>Gioco.h</i>	244
5.10.7	<i>StatoFisico.h</i>	244
5.10.8	<i>StatoFisico.cpp</i>	245
5.10.9	<i>Palestra.h</i>	246
5.10.10	<i>Scheda.h</i>	246
5.11	SALA GIOCHI.....	247
5.11.1	<i>Aggiornamento mappa</i>	247
5.11.2	<i>Aggiornamento dei vocaboli</i>	248
5.11.3	<i>Aggiornamento azioni</i>	248
5.11.4	<i>Aggiornamento oggetti</i>	248
5.11.5	<i>Aziona gioca slot-machine</i>	249
5.11.6	<i>Metodo load() e save()</i>	252
5.12	IMPLEMENTAZIONE BIBLIOTECA.....	253
5.12.1	<i>Aggiornamento mappa</i>	253
5.12.2	<i>Aggiornamento dei vocaboli</i>	254
5.12.3	<i>Aggiornamento azioni</i>	254
5.12.4	<i>Aggiornamento oggetti</i>	254
5.12.5	<i>Dichiarazione azioni</i>	254
5.12.6	<i>Implementazione metodi per funzionamento biblioteca</i>	258

5.12.7	<i>ChangeLog</i>	263
5.13	IMPLEMENTAZIONE BARI – ROMA – PISA	268
5.13.1	<i>Mappa.nav</i>	269
5.13.2	<i>Descrizioni(cartella)</i>	269
5.13.3	<i>Astro.cpp</i>	269
5.13.4	<i>Astro.h</i>	277
5.13.5	<i>Gioco.cpp</i>	277
5.13.6	<i>Gioco.h</i>	279
5.13.7	<i>Luogo.cpp</i>	279
5.13.8	<i>Luogo.h</i>	279
5.13.9	<i>Oggetti.cpp</i>	280
5.13.10	<i>Oggetti.h</i>	280
5.13.11	<i>Oggetto.h</i>	280
5.13.12	<i>Oggetto.cpp</i>	280
5.13.13	<i>Veicolo.h</i>	281
5.13.14	<i>Veicolo.cpp</i>	282
5.13.15	<i>Bagagliaio.h</i>	283
5.13.16	<i>Bagagliaio.cpp</i>	284
5.13.17	<i>Autobus.h</i>	286
5.13.18	<i>Autobus.cpp</i>	286
5.13.19	<i>Automobile.h</i>	287
5.13.20	<i>Automobile.cpp</i>	287
5.14	MODIFICA CODICI LUOGHI – PALAGIANO MARCELLO	288
5.14.1	<i>Modifica file “Mappa.nav”</i>	288
5.14.2	<i>Modifica file “MappaOsservatorio.nav”</i>	289
5.14.3	<i>Modifica file “MappaAliena.nav”</i>	290
5.14.4	<i>Modifica file “Trasporti.nav”</i>	291
5.14.5	<i>Modifica righe di codice</i>	291
5.15	IMPLEMENTAZIONE MECCANICO	296
5.15.1	<i>Aggiunta del file “Batteria.h”</i>	297
5.15.2	<i>Aggiunta del file “Batteria.cpp”</i>	297
5.15.3	<i>Aggiunta del file “Attivita.h”</i>	298
5.15.4	<i>Aggiunta del file “Attivita.cpp”</i>	299
5.15.5	<i>Modifica al file “Gioco.cpp”</i>	300

5.15.6	<i>Modifica al file "Gioco.h"</i>	301
5.15.7	<i>Modifica al file "Astro.h"</i>	302
5.15.8	<i>Modifica al file "Astro.cpp"</i>	303
5.15.9	<i>Modifica al file "Mappa.nav"</i>	315
5.16	CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”	316
5.16.1	<i>Verbo “ruba”</i>	316
5.16.2	<i>Verbo “rubati”</i>	316
5.16.3	<i>Personaggio “carabiniere”</i>	316
5.16.4	<i>Luogo “caserma”</i>	317
5.17	ZAINO E VALIGIA	317
5.17.1	<i>Astro.h</i>	317
5.17.2	<i>Astro.cpp</i>	317
5.17.3	<i>Gioco.h</i>	324
5.17.4	<i>Gioco.cpp</i>	325
5.17.5	<i>Mappa.h</i>	339
5.17.6	<i>Interfaccia.cpp</i>	339
5.17.7	<i>Oggetti.h</i>	340
5.17.8	<i>Oggetti.cpp</i>	340
5.17.9	<i>Oggetto.h</i>	342
5.17.10	<i>Oggetto.cpp</i>	343
5.17.11	<i>Zaino.h</i>	343
5.17.12	<i>Valigia.h</i>	343
6	STRUTTURE INTERCAMBIABILI.	344
6.1	LISTA	344
6.1.1	<i>Lista con Puntatore</i>	344
6.1.2	<i>Lista con Vettore sequenziale</i>	350
6.1.3	<i>Lista con puntatori doppi</i>	353
6.2	LISTA ORDINATA	357
6.2.1	<i>Lista Ordinata con vettore:</i>	357
6.2.2	<i>Lista Ordinata con puntatori:</i>	359
6.3	PILA.....	360
6.3.1	<i>Pila con puntatori:</i>	360
6.4	CODA.....	362
6.4.1	<i>Coda con lista</i>	363

6.4.2	<i>Coda con puntatori</i>	365
6.4.3	<i>Coda con Vettore sequenziale</i>	369
6.5	<i>CODA DI PRIORITÀ</i>	370
6.5.1	<i>Coda con priorità con vettore (Heap)</i>	370
6.5.2	<i>Coda con priorità con lista ordinata</i>	376
6.6	<i>ALBERO N-ARIO</i>	385
6.6.1	<i>AlberoNario: realizzazione con alberi binari</i>	385
6.6.2	<i>AlberoNario: realizzazione primofiglio/fratello</i>	395
6.7	<i>INSIEMI</i>	399
6.7.1	<i>Insiemi con lista</i>	399
6.8	<i>DIZIONARIO</i>	403
6.8.1	<i>Dizionario con vettore ordinato</i>	403
6.8.2	<i>Dizionario con lista</i>	406
6.9	<i>GRAFI</i>	408
6.9.1	<i>Matrice di Adiacenza</i>	408
6.9.2	<i>Vettore di Liste di Adiacenza</i>	416

1 TRACCIA

Durante il corso di algoritmi e strutture dati in “Informatica e Comunicazione Digitale” si è esaminato il gioco di avventura “La nave condannata”.

Le varie modifiche commissionate avevano lo scopo di far sviluppare e implementare, all'interno del gioco, varie strutture dati per offrire nuove funzionalità all'utente.

1.1 Modifica Antonio Pastorelli

Il progetto consiste nella modifica del luogo “Museo”, progettando ed implementando un sistema di scelta per lo spostamento dell’utente nel museo, utilizzando le opportune strutture dati.

1.2 MODIFICA ANTONIO BASILE

Il progetto consiste nel modificare il progetto dello studente Del Giudice (progetto base), implementando all'interno del progetto quattro nuovi luoghi, che sono il luogo “scuola” e le rispettive classi “1A”, “2A” e “3A”.

1.3 MODIFICA DEL GIUDICE ANGELO

Il progetto consiste nel modificare il progetto di Galeandro e prenderlo quindi come progetto “base” per l'integrazione di nuove funzionalità contenute nel progetto di Crocco.

L'integrazione consiste nell'inserire due luoghi, assenti nel progetto base, ovvero il luogo “Archivio” e il luogo “Museo”.

Nell'Archivio saranno disponibile dei file da poter leggere a condizione prima di averli scaricati, pertanto si dovrà prima interagire con il “computer” presente all'interno del luogo Archivio stesso, e una volta scaricati saranno disponibili per la lettura.

Nel luogo Museo invece si potrà entrare solo e soltanto se si dispone dell'oggetto “portafoglio” e non solo ma anche di almeno “10 euro” al suo interno, oggetti che potranno essere presi in altri momenti e luoghi all'interno del gioco. Una volta ottenuto l'accesso al museo si potranno percorrere le varie gallerie al suo interno con tante informazioni culturali per ogni galleria.

1.4 MODIFICA MOSCHETTI MARCO

Si richiede di integrare nel progetto base di Galeandro Germano le parti presenti nel progetto di Schirano Giuseppe che mancano, uniformando le strutture dati presenti rendendo intercambiabili le diverse realizzazioni della stessa struttura.

1.5 MODIFICA GERMANO GALEANDRO

Si richiede di individuare ed integrare le modifiche apportate dal progetto di Riccardo Bellanova al progetto D'Andria\Dresda.

Bisogna aggiungere di conseguenza il luogo "Ufficio Postale" e le relative funzionalità tenendo presente delle possibili collisioni con le modifiche apportate da D'Andria\Dresda.

Inoltre viene richiesto di raggruppare in un'unica cartella tutte le strutture intercambiabili presenti nei vari progetti dei colleghi, di verificarne l'intercambiabilità e di documentarne le differenze in caso di strutture con lo stesso tipo di realizzazione.

1.6 MODIFICA D'ANDRIA IVAN – DRESDA CLAUDIO

Aggiungere la funzionalità di dialogo tra alcuni personaggi non giocanti presenti nell'avventure ed il giocatore; tale dialogo è organizzato con un albero n-ario: i nodi sono le frasi che può dire il personaggio, e gli archi sono possibili risposte che può dare il giocatore, ciascuna delle quali porta ad una nuova frase del personaggio; la radice è la frase con cui il personaggio si presenta; ogni volta che si parla con il personaggio, devono essere visualizzate le possibili risposte che il giocatore può selezionare; quando il giocatore incontra nuovamente un personaggio, devono essere visualizzate le possibili risposte dello stesso punto in cui era rimasto l'ultima volta; Ogni personaggio ha un albero di dialogo diverso, che va caricato da file all'inizio del gioco; quando si salva il gioco, per ciascun personaggio va salvato lo stato del dialogo.

1.7 MODIFICA FEDERICA FORTE

Si richiede di integrare al progetto base di Margherita Disabato e quindi all'interno del mondo di gioco il progetto di Giuseppe Prò, ovvero il luogo "sala scommesse" in cui il personaggio potrà svolgere delle scommesse in moneta di gioco ("contanti").

1.8 MODIFICA GIANLUCA VACCA

Il progetto consiste nell'aggiunta del luogo "Auditorium", implementando delle strutture dati adatte.

1.9 MODIFICA SALVATORE VESTITA

Il progetto consiste nel modificare il progetto dello studente Gianluca Vacca (progetto base), integrando in esso le funzionalità mancanti prese dal progetto dello studente Antonio Semeraro (aggiunta del luogo "stazione").

1.10 MODIFICA VINCENZO GIANNUZZO

Il progetto consiste nel modificare il progetto dello studente Salvatore Vestita (progetto base), integrando in esso le funzionalità mancanti prese dal progetto dello studente Antonio Savino (aggiunta del luogo "Palestra").

1.11 MODIFICA CICALA GIACOMO

Si richiede di integrare nel progetto di Mantellini le funzionalità mancanti, prese dal progetto di Stefano Raffa. In particolare verrà integrato il luogo "Sala Giochi".

1.12 MODIFICA SCATIGNA GIANLUCA

Si richiede di integrare al progetto base di Cicala Giacomo il luogo biblioteca presente nel progetto di Sternativo Francesco, in cui il personaggio potrà prendere dei libri in prestito.

1.13 MODIFICA DELLA FOLGORI GRAZIA

Integrare il progetto di Narracci (che ha implementato: autobus, automobile, tre mappe e le chiavi dei rispettivi mezzi) in quello di Tursi.

1.14 MODIFICA PALAGIANO MARCELLO

Si richiede di integrare nel progetto base di Della Folgori Grazia il luogo "Meccanico" presente nel progetto di Fasano Angelo, e di verificare che il comportamento sia coerente con il fatto che si abbia la macchina (ad es., non si può chiedere di cambiare la batteria se non si ha la macchina).

1.15 MODIFICHE CHIARAPPA ROSA

Si richiede di integrare nel progetto “base1909” di Palagiano Marcello:

- un verbo “ruba” che fa rubare al personaggio un oggetto della stanza
- creare un nuovo luogo “caserma”
- integrare un personaggio “carabiniere” che compare casualmente; se trova un oggetto rubato in possesso del personaggio lo porta in un luogo “caserma”

1.16 MODIFICHE LUCERI MATTEO

Si richiede di modificare il progetto base (base 1911 - Rosa Chiarappa) in modo che vengano aggiunte le funzionalità presenti in quello di Mirco Sternativo, nello specifico, l'inserimento di uno zaino ed una valigia.

2 ANALISI

“Adventure with Simunav” è un gioco d'avventura testuale in cui il giocatore impersonerà il comandante dell'astronave “Neutronia” che a causa di una fatale avaria, dovrà salvare l'equipaggio. Per farlo, il giocatore dovrà navigare all'interno dell'ambiente di gioco e, superando ostacoli, risolvendo eventuali enigmi e prendendo oggetti, alcuni necessari per il proseguimento del gioco altri no, dovrà arrivare alla soluzione del gioco. Inizialmente il giocatore dovrà decidere se affrontare l'intera avventura rispondendo agli enigmi che si presenteranno ad ogni mossa effettuata, oppure giocare senza di essi. Tali indovinelli permetteranno di effettuare mosse solo se risolti, in caso contrario bisognerà risolverne un altro prima di compiere un'azione. Il tempo a disposizione per la riuscita dell'impresa è limitato, ma saranno presenti diverse ricompense a seguito di determinate azioni all'interno del gioco, che permetteranno di ricevere una quantità di secondi extra.

Il protagonista potrà spostarsi all'interno del gioco digitando dei comandi: Nord(N), Sud(S), Ovest(W), Est(E); quindi potrà spostarsi all'interno dell'astronave con un navigatore, per orientarsi con più facilità nell'intero gioco, che potrà essere richiamato in qualsiasi momento della partita.

I vari luoghi offrono interazioni uniche, possibilità di guadagnare tempo (ma di perderlo anche).

2.1 MUSEO

Il luogo “Museo” prevede un sistema di movimento all'interno di quest ultimo, dando la possibilità all'utente di scegliere in quali stanze del museo spostarsi e visualizzarne i contenuti informativi, tutto ciò è già implementato senza l'utilizzo di opportune strutture dati. L'obiettivo di questa modifica è quello di progettare ed implementare questo sistema utilizzando una struttura dati che fornisca gli strumenti necessari ad ottimizzarlo.

2.2 SCUOLA E AULE

Si sono aggiunte al gioco quattro nuovi luoghi, la Scuola con le sue rispettive classi, che saranno inseriti in un punto specifico della mappa.

Le azioni che verranno eseguite all'interno della Scuola non hanno finalità per la soluzione del gioco, tuttavia si può incrementare il tempo di gioco, in maniera tale che il giocatore avrà più tempo per arrivare alla soluzione finale.

Nel luogo Scuola è possibile dialogare con la persona “preside”, che farà delle domande, al protagonista di cultura generale. Se la risposta è corretta, il tempo di gioco viene incrementato, altrimenti, il tempo di gioco viene decrementato.

Nella classe 1A è possibile dialogare con il personaggio “maestra Clara”, maestra di storia. Farà un test di storia al giocatore, ad ogni risposta corretta il tempo verrà incrementato e ad ogni risposta errata il tempo verrà decrementato.

Nella classe 2A è possibile dialogare con il personaggio “maestra Mara”, maestra di matematica. Farà un test di matematica al giocatore, ad ogni risposta corretta il tempo verrà incrementato e ad ogni risposta errata il tempo verrà decrementato.

Nella classe 3A il giocatore può risolvere un gioco scritto alla lavagna, il gioco dell’impiccato. Se risolve il gioco, il suo tempo sarà incrementato, viceversa decrementato.

Inoltre in ogni classe ci saranno gli oggetti che formano una classe, dei banchi, delle sedie, delle lavagne, le cartine geografiche appese alle pareti eccetera.

2.3 ARCHIVIO E MUSEO (VECCHIA VERSIONE DEL MUSEO, INSERITA PER COMPLETEZZA)

Nel luogo Archivio saranno presenti dei “file” leggibili ma in principio bloccati in quanto disponibili per la lettura solo dopo averli scaricati dal terminale, infatti nell’Archivio sarà presente il “computer” a cui si potrà accedere attraverso una specifica azione e che darà la possibilità di scegliere quale file scaricare e di scaricarlo, rendendo disponibile così per la lettura il file stesso. Esso conterrà solo delle informazioni per rendere meno banale l’esperienza di gioco. Tuttavia per poter accedere ai file del computer bisognerà attendere il proprio turno e si potrà scegliere se aspettare oppure non aspettare e quindi uscire.

Nel luogo Museo invece il giocatore si troverà difronte alla biglietteria del museo che verificherà se la condizione di entrata è rispettata, in quanto per poter accedere al Museo vero e proprio e quindi alle varie gallerie bisognerà disporre obbligatoriamente dell’oggetto “portafoglio” con almeno “10 euro” al suo interno.

Quando si disporrà di tali oggetti si potrà entrare e anche qui ci si troverà difronte a una fila potendo decidere se aspettare oppure no.

Una volta entrati si potranno visionare quattro Gallerie (A,B,C,D) con varie informazioni culturali inutili però ai fini del completamento del gioco.

2.4 AULA SINGOLA

Nell’aula il giocatore potrà compiere azioni utili o necessarie alla risoluzione del gioco, ma anche azioni inutili, che serviranno solo a rendere più imprevedibile e meno banale l’esperienza di gioco, quali seguire delle lezioni, effettuare ricerche al computer, leggere libri e proiettare messaggi tramite un proiettore.

2.5 UFFICIO POSTALE

Nell’Ufficio postale sarà possibile inviare e ricevere oggetti e si potrà interagire con la Banca.

L’Ufficio Postale esplicherà le sue funzioni tramite uno Sportello telematico che richiederà un documento d’identità per il servizio di ricezione e darà accesso a due nuovi oggetti : Una Lettera con dei suggerimenti e un Pacco con un oggetto misterioso (la chiave del secondo pilota).

Per accedere allo Sportello telematico si dovrà prima attendere il proprio turno in base a una fila randomica di persone. Sarà comunque possibile scegliere se aspettare il proprio turno o ritentare più in là.

Tutte le funzionalità dell’Ufficio Postale non sono comunque necessarie ai fini del completamento del gioco.

2.6 DIALOGHI

Il giocatore potrà dialogare con i vari personaggi non giocanti presenti nell'avventure

2.7 SALA SCOMMESSE

Nel luogo identificato come "Sala scommesse" il giocatore potrà svolgere attività che gli permetteranno di incrementare o diminuire la moneta di gioco ("contanti"), attraverso delle scommesse su una simulazione di corsa delle astronavi.

La sala si trova a nord della banca, in modo da permettere al giocatore di usufruire dei servizi offerti dalla banca per depositare o prelevare altro denaro.

Inoltre non è un luogo a pagamento, questo per dare la possibilità di entrare indipendentemente se si possiede denaro, anche solo per guardare il simulatore, così da dare l'idea di ciò che tale oggetto permette di fare.

2.8 SIMULATORE

All'interno del luogo "Sala scommesse" è stato inserito un oggetto denominato "Il Simulatore" (oggetto non trasportabile).

Esso sarà utile al giocatore per effettuare scommesse sul gioco "corsa delle astronavi", su cui puntare in moneta di gioco ("contanti"), e provare a triplicare la propria somma.

Si è scelto di utilizzare come metodo di pagamento di gioco i contanti in maniera da permettere al personaggio di ricevere un qualcosa di versatile da poter utilizzare anche in altri luoghi presenti nel mondo di gioco.

Si è pensato di creare una coda per l'accesso al simulatore in maniera da emulare situazioni presenti nel mondo reale in cui più persone attendono per utilizzare lo stesso macchinario di gioco.

2.9 AUDITORIUM

L'auditorium è un luogo in cui il protagonista può interagire con svariati oggetti, fra cui un jukebox antico e un pannello di controllo delle luci.

2.10 STAZIONE

In base alla modifica richiesta è stato aggiunto il luogo "Stazione", in cui sono presenti i seguenti oggetti: panchina, vecchio giornale, treno, biglietteria. Questo luogo è stato pensato e realizzato al fine di "distrarre" ed "ostacolare" il giocatore, infatti l'esecuzione delle azioni in tale luogo non portano alla soluzione del gioco.

Il giocatore potrà, tramite la biglietteria, acquistare biglietti spendendo una quantità di denaro non rimborsabile. Il treno però, a causa di un guasto, sarà impossibilitato a partire e, di conseguenza, l'utente avrà speso il proprio tempo ed il proprio denaro inutilmente!

In questo luogo è presente anche un giornale, al quale è stato attribuito l'aggettivo "vecchio", in quanto nel momento in cui il giocatore proverà a leggerlo sarà impossibilitato a farlo, dato che il giornale risulta frammentato.

Nella stazione è anche presente una panchina. Il giocatore potrà sedersi sulla panchina o alzarsi da questa, ma non potrà effettuare alcuna azione stando seduto!

Infatti, per essere corenti con l'obiettivo del luogo, al giocatore, una volta seduto, verrà sottratto 1 punto al tempo.

2.11 BIBLIOTECA

Nella Biblioteca il giocatore potrà prendere in prestito dei libri che potrà consultare in qualunque momento all'interno del gioco.

Si è pensato di inserire il nuovo luogo salendo le scale nella scuola, in modo da permettere agli studenti di avere un luogo in cui andare a leggere dei libri inerenti all'astronave.

Inoltre, verrà inserito il sotto luogo "Vetrina" come luogo in cui è possibili visualizzare i libri disponibili ed eventualmente prenderli in prestito.

2.12 BARI – ROMA – PISA

Nel progetto di Tursi (progetto base), sono stati implementati i seguenti elementi, implementati da Narracci (progetto da integrare):

- Autobus
- Automobile
- Chiave automobile
- Ticket Bus

Per accedere ai mezzi ed interagire con essi il giocatore dovrà prendere rispettivamente la chiave e il ticket, situati nella “cabina del secondo pilota”, per automobile e autobus.

Inoltre, la mappa del progetto di base è stata ampliata aggiungendo tre nuovi luoghi raggiungibili partendo dalla “stazione di servizio” a piedi o utilizzando i mezzi. Nel caso di mancato utilizzo dei mezzi per queste destinazioni il giocatore verrà avvisato con <<Impieghi più tempo a piedi>> e gli verrà scalato maggiore tempo.

I nuovi oggetti, luoghi e veicoli non sono utili alla storia e al completamento del gioco, sono una distrazione in più per il giocatore.

2.13 MECCANICO

Nel luogo “Meccanico” l'avventuriero potrà interagire con gli oggetti presenti, ma non tutte le azioni compiute sono utili o indispensabili ai fini della soluzione del gioco. Nel luogo “Meccanico” sarà possibile interagire con diversi oggetti quali, ad esempio, il banco da lavoro, un diario contenente alcuni appunti, delle batterie oppure un computer. Inoltre, sarà presente la figura del *meccanico* che potrà, secondo la volontà del giocatore, cambiare la batteria dell'*automobile*.

2.14 VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”

Il progetto consiste nell'integrare un verbo “**ruba**” che farà rubare al giocatore alcuni oggetti presenti in determinate stanze del gioco. Sarà creato un inventario che verrà richiamato col verbo “**rubati**” che elencherà tutti gli oggetti rubati. Verrà inserito un personaggio “**carabiniere**” che, quando incontrerà il giocatore, lo perquisirà controllando se in suo possesso avrà degli oggetti rubati: se non li trova il giocatore potrà continuare a giocare, altrimenti verrà portato in un luogo “**caserma**” creato appositamente.

2.15 ZAINO E VALIGIA

Le funzionalità da implementare nel progetto di base sono:

L'inserimento di oggetti, uno **zaino** o in una **valigia**, ossia dei contenitori che permettono all'utente di trasportare determinati oggetti (come casco, tuta, camice o manuale). È possibile, inoltre, decidere se portare questi oggetti con sé o metterli nella valigia o nello zaino.

Lo zaino e la valigia hanno dei limiti di peso trasportabile. La posizione di questi due contenitori sarà fissa: ad ogni nuova partita si troveranno sempre nello stesso luogo, con la possibilità di scegliere se utilizzarli o meno. Non sarà possibile trasportare entrambi gli oggetti contenitori, bisognerà scegliere solo uno dei due.

Nella valigia si possono inserire e prelevare quanti oggetti si vuole, rimanendo nei limiti del peso massimo consentito, invece nello zaino si potranno sì mettere quanti oggetti si vuole rimanendo nei limiti del peso, ma si potrà prendere solo il primo di questi.

Ciò comporta che, nel caso in cui si voglia esplorare tutto lo zaino, bisognerà togliere tutti gli oggetti presenti. L'utilizzo dello zaino o della valigia è a discrezione dell'avventuriero che, in caso ce l'abbia, può decidere se trasportare con sé gli oggetti o metterli nel contenitore.

3 PROGETTAZIONE

3.1 MUSEO

La struttura dati scelta per risolvere il problema è il grafo orientato. Ogni nodo del grafo rappresenterà una stanza del museo mentre ogni arco rappresenterà la possibilità di spostarsi da una stanza all'altra.

Una volta entrato nel museo l'utente verrà posizionato nel nodo che rappresenta l'entrata del museo e potrà muoversi di stanza in stanza selezionandola in un elenco di movimenti disponibili.

Questo elenco verrà creato ogni volta che l'utente dovrà scegliere dove spostarsi grazie alla lista dei nodi adiacenti al nodo corrente.

Inoltre prima dell'elenco delle stanze adiacenti, verrà visualizzato a schermo il contenuto informativo della stanza corrente.

In ogni stanza del museo verrà aggiunto un elemento all'elenco dei movimenti disponibili che rappresenta l'“uscita del museo”, per facilitare l'utente nel caso in cui volesse uscire dal museo da qualsiasi posizione. Questa scelta è stata fatta alla luce dell'ipotesi di una futura crescita del museo, in tal caso potrebbe diventare difficoltoso trovare la strada per uscire, quindi verrà implementata questa scelta “uscita del museo” che posizionerà l'utente all'entrata del museo e poi lo farà uscire, mantenendo la possibilità di rientrare e riprendere la sua visita dall'entrata del museo come se fosse la prima volta.

3.2 SCUOLA E AULE

3.2.1 Luogo Scuola

Il luogo “Scuola” è accessibile dalla cabina di pilotaggio, procedendo a nord di essa.

Il giocatore incontrerà “il preside” che gli può formulargli una serie di domande.

Il preside chiederà il nome del giocatore e se vuole rispondere a delle domande per la risoluzione di un cruciverba.

Inoltre ci sarà l'oggetto cartina galattica che rappresenta il “Sistema Solare”.

3.2.2 Luogo Classe 1A

La “Classe 1A” è accessibile dal luogo “Scuola” e si troverà a est di essa.

In questa Aula troveremo gli oggetti “banchi”, “sedie” e “cattedra” che non sono trasportabili.

Il giocatore potrà dialogare con il personaggio “maestra Clara”, maestra di Storia che proporrà al protagonista un test con tre domande di storia.

Nell'aula è presente una cartina geografica che rappresenta “la penisola italiana”, non si può essere trasportata e quindi fissa nell'aula.

3.2.3 Luogo Classe 2A

La “Classe 2A” è accessibile dal luogo “Scuola” e si troverà a nord di essa.

In questa Aula troveremo gli oggetti “banchi”, “sedie” e “cattedra” che non sono trasportabili.

Il giocatore potrà comunicare con il personaggio “maestra Mara”, maestra di matematica che proporrà al protagonista un test con la risoluzione di espressioni matematiche. Le espressioni saranno tre.

Inoltre ci sarà l'oggetto “Registro” dove sono annotate le presenze e le assenze degli alunni. L'oggetto “Registro” può essere trasportato e consultato.

3.2.4 Luogo Classe 3A

La “Classe 3A” è accessibile dal luogo “Scuola” e si troverà a ovest di quest’ultima.

In questa Aula troveremo gli oggetti “banchi”, “sedie” e “cattedra” che non sono trasportabili.

Ci sarà l'oggetto “lavagna” non trasportabile raffigurante il gioco dell'impiccato. Se il gioca verrà risolto, in maniera corretta il tempo sarà incrementato.

3.3 ARCHIVIO E MUSEO (VECCHIA VERSIONE DEL MUSEO)

3.3.1 Luogo Archivio

Il luogo Archivio è accessibile dall'aula infatti si troverà a sud della stessa.

Il giocatore si troverà davanti ad una serie di file e ad un computer, l'azione di lettura dei file non sarà disponibile fin quando non saranno scaricati dal terminale infatti il giocatore dovrà applicare l'azione per utilizzare il computer.

Tuttavia per poterlo realmente usare per scaricare i file dovrà attendere il suo turno in quanto ci sarà una fila di n persone generate in maniera casuale da 1 a 10. I componenti della fila rappresentano un' unità di tempo da perdere per poter utilizzare il computer, verrà perciò data la possibilità di scegliere se aspettare oppure no. Nel primo caso ci si troverà difronte quindi alla scelta dei file da scaricare, nel secondo caso invece si uscirà dal computer e si tornerà indietro.

3.3.2 File

I documenti stampati potranno essere visualizzabili solo nel luogo Archivio infatti non rappresentano degli oggetti trasportabili. Essi contengono informazioni utili allo svolgimento del gioco (come il funzionamento di alcune parti dell'astronave) o dettagli sulla storia. Il giocatore potrà possedere una sola copia dello stesso e, nel caso si tenti di stamparne un'altra, verrà visualizzato un avviso che avverte l'utente di possederne già una. Nel caso si tenti di stampare lo stesso documento più volte, pur non consentendo l'operazione, verrà comunque scalato del tempo.

3.3.3 Museo

Il luogo Museo sarà accessibile dal luogo Archivio infatti sarà presente a sud dello stesso.

Il giocatore si troverà difronte alla biglietteria del Museo, se tenterà di eseguire l'azione di entrata nello stesso il sistema controllerà che abbia il permesso necessario.

Questo permesso è dato dal possedere oppure no l'oggetto “portafoglio” con almeno “10 euro” al suo interno, infatti in caso di mancanza di uno dei due il sistema stamperà dei messaggi per far capire al giocatore che per poter entrare ha bisogno di quei due oggetti.

Una volta ottenuti quei due oggetti, che sono presenti in altri punti diversi dal Museo all'interno del gioco, il biglietto sarà acquistato e ci si troverà difronte ad una fila di n persone generate casualmente da 1 a 10 con la possibilità di decidere se aspettare oppure no. Nel primo caso il giocatore si troverà difronte alle varie gallerie, nel secondo si uscirà dal museo e si ritirerà difronte alla biglietteria.

Se si decidere di aspettare quindi si entrerà nel Museo con la possibilità di scegliere quale galleria visitare tra quelle disponibili ovvero Galleria A, Galleria B, Galleria C, o Galleria D.

Ognuna di queste permetterà al giocatore di scegliere a sua volta degli argomenti culturali di cui potrà prendere visione.

3.4 AULA SINGOLA

In base alla modifica richiesta, si aggiungerà come nuovo luogo un'aula all'estremità sud del corridoio, in cui, tra le possibili attività che il giocatore potrà compiere, quelle utili alla risoluzione del gioco saranno:

- Seguire delle lezioni: porterà il giocatore alla conoscenza di informazioni utili o necessarie alla risoluzione del gioco.
- Leggere dei libri: permetteranno al giocatore di recuperare 5 unità di salute una volta sola nel gioco e quindi si dovrà decidere se usufruirne subito o in un altro momento dell'avventura.

3.4.1 Computer

Nell'aula sarà presente un computer attraverso il quale sarà possibile effettuare ricerche su argomenti casuali in qualsiasi momento del gioco. Tali ricerche però non saranno utili allo svolgimento dello stesso, infatti il computer è un mezzo per depistare il giocatore e fargli perdere tempo prezioso. Questo oggetto non può essere trasportato.

3.4.2 Sedie

Nell'aula saranno presenti delle sedie sulle quali il giocatore potrà sedersi in qualsiasi momento del gioco. Come per il computer, anche le sedie non saranno utili allo svolgimento del gioco, infatti sono un mezzo per depistare il giocatore e fargli perdere tempo prezioso. A differenza del computer però, le sedie potranno essere trasportate.

3.4.3 Proiettore

Nell'aula sarà presente un proiettore che il giocatore potrà accendere o spegnere a seconda della sua volontà. Una volta acceso, sarà proiettato un messaggio utile, che servirà a far capire al giocatore che la lettura potrà portargli dei benefici, invogliandolo così a leggere i libri presenti nell'aula. Come per il computer, anche questo oggetto non potrà essere trasportato.

3.4.4 Libri

Nell'aula saranno presenti dei libri con i quali il giocatore potrà recuperare 5 unità di salute una volta letti. Questo recuperò però potrà avvenire solo una volta nel corso del gioco.

Tra i libri presenti nell'aula, solamente uno conterrà il kit medico che consentirà al giocatore di usufruire del beneficio.

L'aumento di salute avverrà solo nel momento in cui lo stato di salute del giocatore sarà minore o uguale a 95/100; infatti se lo stato di salute dovesse essere maggiore di 95, il giocatore usufruirà lo stesso del kit senza trarne alcun beneficio, perdendo così l'occasione di utilizzarlo nel momento di reale bisogno.

Se l'utilizzo corretto del kit medico dovesse avvenire nel momento in cui il giocatore sarà ferito, l'incremento di salute potrà incidere direttamente sul tempo necessario per lo svolgimento delle azioni del gioco. Come per le sedie, anche i libri potranno essere trasportati.

3.4.5 Lezione

Nel progetto da integrare è stato inserito un oggetto "lezione in corso".

Nel momento in cui il giocatore effettuerà il comando "segui lezione" potrà decidere quale lezione seguire. Tra le lezioni che si potranno scegliere, solo alcune saranno utili ai fini del gioco, mentre altre faranno solamente perdere del tempo prezioso al giocatore.

Ogni qual volta si vorrà seguire una lezione si perderanno 5 unità di vita (unità di tempo) sia che la lezione sia utile, sia che essa sia inutile.

3.5 UFFICIO POSTALE

L'Ufficio Postale sarà il 24esimo luogo di questa versione dell'Adventure. Sarà raggiungibile a Sud attraverso l'Ufficio\Deposito e sarà possibile accedere alle sue funzioni tramite l'oggetto "Sportello Telematico". È stata quindi creata una nuova classe "Ufficio Postale" sul modello del "luogoufficio" già precedentemente implementato.

La classe si avvarrà di una funzione random nella "cstdlib" utilizzata per generare una fila randomica di persone (da 0 a 10 compresi) ogni volta che si accederà allo Sportello Telematico.

3.6 DIALOGHI

1. Incontro personaggio

1.1. Caricamento stato domanda corrente:

- Se incontro il personaggio per la prima volta stampa presentazione e prima domanda
- Se ho già iniziato un dialogo con il personaggio stampo l'ultima domanda con le proprie possibilità di risposte che erano state proposte precedentemente

2. Dialogo

1.2. fino a quando non si decide di interrompere il dialogo o il personaggio termina le domande da porre:

- 2.1.1.Stampo domanda e possibilità di risposte allo stato corrente
- 2.1.2.Leggo risposta
- 2.1.3.Aggiorno stato dialogo

3. Fine dialogo

1.3. Dialogo terminato per domande esaurite?

- Se sì stampo ultimo messaggio di saluto e riporto lo stato del dialogo al valore di partenza
- Se no salvo lo stato corrente del dialogo

La classe Dialogo avrà come attributi:

- il nome del personaggio a cui è associato il dialogo;
- lo stato del dialogo ovvero il codice dell'ultima domanda posta all'utente;
- un albero n-ario, i cui nodi saranno di tipo Domanda, che conterrà il dialogo.

La classe Domanda avrà come attributi:

- il codice domanda che sarà costruito in base alla posizione del nodo all'interno dell'albero del dialogo (es. radice: codice=1, il primo fratello del primo figlio della radice: codice=12);
- una stringa che conterrà la domanda con le possibili risposte che l'utente può dare.

Quando si incontra un personaggio per la prima volta il programma visualizza a video la domanda avente codice 1 che corrisponde alla radice dell'albero del dialogo.

L'utente risponderà con 1, 2 o 3 a seconda di quale risposta voglia dare.

Nel caso risponda con 0, verrà interrotto il dialogo.

La risposta verrà concatenata allo stato attuale del dialogo diventando il nuovo stato dello stesso; se per esempio alla prima domanda viene data la risposta 2, il nuovo stato sarà 12.

In seguito verrà visualizzata a video la domanda avente per codice lo stato del dialogo; nell'esempio precedente verrà stampata la domanda avente codice 12 (che sarà il primo fratello del primo figlio della radice).

Il procedimento proseguirà fino a quando non si arriverà a una foglia dell'albero del dialogo, oppure fino a quando l'utente non risponda con 0 e in tal caso verrà salvato lo stato in cui si trova il dialogo.

Quando si incontrerà in seguito lo stesso personaggio verrà stampata a video la domanda avente codice corrispondente allo stato in cui si era fermato il dialogo in precedenza.

3.7 SALA SCOMMESSE E SIMULATORE

All'interno del luogo di gioco "Sala scommesse", è stato aggiunto l'oggetto "Simulatore", oggetto che farà perdere tempo al giocatore, ma in caso di vittoria gli consentirà di guadagnare denaro di gioco sottoforma di contanti.

Per aggiungere l'oggetto, alla lista degli oggetti, si è dovuto assegnare un etichetta univoca, un codice univoco, ed un codice mappa negativo poiché tale oggetto non è trasportabile.

ETICHETTA	CODICE OGGETTO	CODICE MAPPA
Un simulatore	58	-23

Successivamente all'aggiunta dell'oggetto nel gioco, è necessario inserire "simulatore" tra i vocaboli del gioco.

ETICHETTA	CODICE OGGETTO
Simulatore	58

3.7.1 Azioni

Le azioni inserite per far interagire il nostro personaggio con il luogo "Sala scommesse" cercano di rispecchiare una situazione di vita reale.

Il personaggio potrà:

- Guardare l'oggetto simulatore, che permetterà di venire a conoscenza delle operazioni possibili da compiere con il simulatore.
- Giocare al simulatore, che permetterà dopo aver fatto una eventuale coda, in attesa che il simulatore si liberi, di poter puntare su una delle astronavi in gara e provare a triplicare la somma puntata. Ogni giocata decrementerà o aumenterà il proprio credito contante, e decrementerà il tempo di gioco di una unità a giocata.

3.7.2 Inserimento Azione "Guarda simulatore"

Per dare idea al personaggio dell'uso dell'oggetto simulatore senza necessariamente giocarci, è stata implementata l'azione guarda simulatore, che permette di scoprire ciò che si può fare giocandoci.

È stato utilizzato il vocabolo guarda già implementato precedentemente con codice 10:

ETICHETTA	CODICE IDENTIFICATIVO
Guarda	10

L'azione potrà essere richiamata, seguita dal vocabolo "simulatore" precedentemente descritto.

3.7.3 Inserimento Azione "Avvia simulatore" e "Gioca simulatore"

Per permettere al personaggio di utilizzare il simulatore per effettuare le proprie scommesse sono state implementate le due azioni "avvia simulatore" e "gioca", che permettono di accedere alle funzioni di gioco.

ETICHETTA	CODICE IDENTIFICATIVO
Gioca	44
Avvia	14
Usa	29

Le azioni "Avvia" e "Usa" potranno essere richiamate, seguite dal vocabolo "simulatore", mentre l'azione "Gioca" potrà essere richiamata da sola.

3.8 AUDITORIUM

Nella costruzione dell'auditorium si è pensato a quali oggetti potessero esserci. L'auditorium è stato pensato come luogo sia di spettacoli, sia di seminari.

Si è deciso di aggiungere una finestra sigillata, una scacchiera appoggiata su una cattedra, dei strumenti musicali (sono 3 e possono essere suonati in base ad una scelta richiesta), un proiettore rotto, uno schermo, un microfono, un jukebox antico e un pannello di controllo delle luci.

Tutti gli oggetti presentano un'interazione semplice (solitamente un messaggio), eccetto *jukebox* e il *pannello di controllo delle luci* che presentano un'interazione più complessa.

Il jukebox è stato realizzato utilizzando una lista come struttura dati e il pannello di controllo, invece, è stato realizzato utilizzando un insieme di luci accese, per tenere traccia di quali luci sono state accese nell'auditorium.

Inoltre si è deciso anche di tenere traccia delle canzoni ascoltate dall'utente ad ogni interazione col Jukebox, quindi tutte le canzoni che vengono riprodotte vengono inserite in una coda. Grazie a questa struttura dati, verrà restituito all'utente un elenco delle canzoni ascoltate in ordine cronologico.

Il jukebox viene inizializzato con 10 canzoni predefinite (dei vecchi classici) ad ogni avvio della partita; il giocatore legge che brano è in riproduzione, chi è l'artista e di quale anno è il brano.

Può anche decidere di tornare alla prima canzone del jukebox antico, di andare avanti di una canzone (nel caso fosse quella finale, il jukebox ritornerà a quella iniziale) o di andare al brano precedente (se il brano attuale è il primo, rimarrà in produzione tale brano poiché il jukebox, essendo antico, non presenta una funzione di tornare al brano finale; questa funzione può essere implementata facilmente se si vuole rendere il jukebox un po' più moderno).

Si è scelto di utilizzare una lista poiché riproduce fedelmente il comportamento di un Jukebox antico.

Ad ogni riproduzione, il brano viene inserito nella Playlist.

Il pannello di controllo, invece, rende possibile accendere le luci dell'auditorium; accendere una luce significa inserire quella determinata luce nell'insieme delle Luci Accese; ogni luce è un oggetto a sé che viene identificato in base alla sua posizione (es: "Destra Avanti").

Se tutte le luci sono spente (quindi l'insieme è vuoto) uscirà un messaggio di luci spente, altrimenti comparirà un messaggio dove vengono elencate le luci accese (ovvero che fanno parte dell'insieme delle luci accese).

Il giocatore, premendo dei tasti numerati, può spegnere o accendere le varie luci ad ogni iterazione col pannello.

L'insieme permette di tenere traccia solamente delle luci accese nell'auditorium, ignorando quelle spente; se una luce non appartiene all'insieme delle luci accese, allora è spenta, senza dover scorrere un eventuale struttura dato che memorizzi tutte le luci presenti nel gioco.

In futuro è possibile ampliare l'oggetto luce (che ora prevede solamente la posizione nella stanza) ed estendere l'insieme (es. si tiene traccia di tutte le luci dell'astronave).

3.9 STAZIONE

In base alla modifica richiesta verrà aggiunto il luogo "Stazione", in cui saranno presenti i seguenti oggetti:

- Panchina
- Vecchio Giornale
- Treno
- Biglietteria

Questo luogo è stato pensato e realizzato al fine di "distrarre" ed "ostacolare" il giocatore, infatti l'esecuzione delle azioni in tale luogo non portano alla soluzione del gioco.

Il giocatore potrà, tramite la biglietteria, acquistare biglietti spendendo una quantità di denaro non rimborsabile. Il treno però, a causa di un guasto, sarà impossibilitato a partire e, di conseguenza, l'utente avrà speso il proprio tempo ed il proprio denaro inutilmente!

In questo luogo è presente anche un giornale, al quale è stato attribuito l'aggettivo “vecchio”, in quanto nel momento in cui il giocatore proverà a leggerlo sarà impossibilitato a farlo, dato che il giornale risulta frammentato.

Nella stazione è anche presente una panchina. Il giocatore potrà sedersi sulla panchina o alzarsi da questa, ma non potrà effettuare alcuna azione stando seduto!

Infatti, per essere corenti con l'obiettivo del luogo, al giocatore, una volta seduto, verrà sottratto 1 punto al tempo.

3.10 PALESTRA

In base alla modifica richiesta verrà aggiunto quindi il luogo “Palestra”, in cui saranno presenti i seguenti oggetti:

- Panca per gli addominali
- Tapis Roulant
- Terminale Informativo
- Schede di allenamento

Questo luogo è stato realizzato al fine di “migliorare” le caratteristiche fisiche del giocatore, in particolare di “costituzione” e “resistenza”.

- **Costituzione:** ridurre la probabilità di ricevere feriti
- **Resistenza:** ridurre il tempo necessario per muoversi all'interno della nave

Il giocatore avrà vari oggetti tra cui scegliere in questo luogo. Il “terminale informativo”, mostrerà a schermo delle informazioni utili sul luogo, fornendo una spiegazione generale sugli oggetti presenti nella palestra.

La “panca” consente di migliorare la costituzione, mentre il “Tapis roulant” permette di incrementare la resistenza. Ma più tempo passiamo sugli attrezzi più il tempo rimanente reale all'interno del gioco, diminuirà.

La scheda permette di stabilire una sequenza di esercizi fra panca e tapis roulant e di assegnarne per ognuno una durata.

3.11 SALA GIOCHI

Come già accennato, è stato inserito un nuovo luogo, la sala giochi. All'interno di esso sarà possibile tentare la fortuna alla slot machine. La fortuna viene intesa come evento favorevole in un insieme di eventi. In particolare, viene sfruttato il concetto di probabilità, in cui solo 1 evento su 100 è favorevole.

Considerando, però, che i premi in palio sono di 5 tipi (5, 10, 20, 50, 100 euro) e che la probabilità di vincere una somma alta è minore rispetto a quella di vincerne una bassa, è stato stilato un piano:

- Probabilità di vincere 5 euro: 16 eventi su 100;
- Probabilità di vincere 10 euro: 8 eventi su 100;
- Probabilità di vincere 20 euro: 4 eventi su 100;
- Probabilità di vincere 50 euro: 2 eventi su 100;
- Probabilità di vincere 100 euro: 1 evento su 100.

Gli eventi favorevoli che ci permetteranno di vincere saranno 31 su 100.

È stato previsto che essa debba avere un numero limitato di banconote al suo interno, in modo da rendere l'idea più realistica (prima o poi la *slot-machine* finirà il denaro al suo interno).

Per fare ciò, è stato inserito al suo interno:

- 1 banconota da 100 euro;
- 2 banconote da 50 euro;
- 4 banconote da 20 euro;
- 8 banconote da 10 euro;
- 16 banconote da 5 euro.

Anche le banconote come gli eventi favorevoli saranno quindi 31.

Per interagire il giocatore guarda la Slot-machine, e potrà verificare prima il numero e quindi la disponibilità di ciascuna banconote al suo interno: comparirà la scritta “DISPONIBILE” oppure “NON DISPONIBILE” accanto ad ogni taglio prima di cominciare a giocare.

3.11.1 Slot machine

Come prima cosa, la *slot-machine* farà apparire a video un messaggio con la disponibilità delle banconote (le quali ricordiamo non essere infinite) e la scelta di poter giocare o uscire.

Dopo di che, estrarrà un numero casuale (da 1 a 100) e verificherà se quest'ultimo sarà un numero vincente o meno, controllando che esso appartenga all'insieme.

Per ultimo verrà inserita anche la possibilità di rigiocare nuovamente senza dover uscire dal gioco.

L'oggetto slot machine è stato realizzato con la struttura insieme (con vettore caratteristico). La realizzazione di questa struttura sfrutta l'accesso diretto ed è immediato il reperimento dell'elemento cercato. È stato creato un vettore di 100 elementi (rappresentanti i 100 eventi), tutti impostati a falso ad eccezione delle prime 31 celle (che rappresentano i 31 eventi favorevoli) che per semplicità saranno impostate a vero.

Se il numero estratto corrisponderà ad una cella impostata a vero, allora l'esito del gioco risulterà positivo, altrimenti risulterà negativo.

Gli oggetti banconote saranno inserite nell'Insieme nelle seguenti posizioni:

- dalla posizione 1 alla posizione 16 - banconote da 5 euro
- dalla posizione 17 alla posizione 24 - banconote da 10 euro
- dalla posizione 25 alla posizione 28 - banconote da 20 euro
- dalla posizione 29 alla posizione 30 - banconote da 50 euro
- nella posizione 31 banconote da 100 euro.

3.12 BIBLIOTECA

Nella Biblioteca il giocatore potrà prendere in prestito dei libri che potrà consultare in qualunque momento all'interno del gioco.

Si è pensato di inserire il nuovo luogo salendo le scale nella scuola, in modo da permettere agli studenti di avere un luogo in cui andare a leggere dei libri inerenti all'astronave.

Inoltre, verrà inserito il sotto luogo "Vetrina" come luogo in cui è possibili visualizzare i libri disponibili ed eventualmente prenderli in prestito.

3.13 BARI – ROMA – PISA

Sono stati aggiunti i seguenti oggetti:

- *Chiave dell'automobile*: necessaria per interagire con l'automobile, si trova nella cabina del secondo pilota.
- *Ticket bus*: necessario per interagire con l'autobus, si trova nella cabina del secondo pilota.
- *Autobus*: è un veicolo accessibile dal giocatore mediante l'utilizzo del ticket apposito. Il giocatore può “prendere” l'autobus e raggiungere Bari, Roma e Pisa.
- *Automobile*: è un veicolo accessibile dal giocatore mediante l'utilizzo della chiave apposita. Il giocatore può “prendere” l'automobile e raggiungere Bari, Roma e Pisa. Inoltre, essa è dotata della funzione aggiuntiva che permette di depositare/prelevare gli oggetti all'interno del bagagliaio. Con il comando “guarda automobile”, il sistema dirà al giocatore che l'auto è dotata di un bagagliaio utilizzabile.

3.14 MECCANICO

Gli oggetti disponibili all'interno del luogo “Meccanico” sono:

- *Un banco da lavoro*: oggetto con cui è possibile interagire tramite il comando “guarda”. Viene, quindi, visualizzata una descrizione di quello che l'avventuriero vede sul banco da lavoro.
- *Un pezzo di ricambio per l'astronave*: oggetto con cui è possibile interagire tramite il comando “guarda”. Viene, quindi, visualizzata una descrizione di quello che l'avventuriero vede relativamente al pezzo di ricambio;
- *Una pila di batterie*: oggetto con cui è possibile interagire tramite il comando “guarda”. L'avventuriero può osservare modello e stato di carica di alcune batterie disposte una sopra l'altra. Qualora lo stato di carica della batteria analizzata sia inferiore al 50%, può scegliere se scartare la batteria inserendola nella pila delle *batterie scariche* oppure rimetterla al proprio posto dopo aver finito di esaminare le altre restanti;
- *Delle batterie scariche*: oggetto con cui è possibile interagire tramite il comando “guarda”. Le batterie scariche compaiono nel luogo “Meccanico” solo quando viene esaminata la *pila di batterie* e solo se il giocatore decide di scartarne alcune;
- *Una chiave_inglese*: oggetto con cui è possibile interagire tramite il comando “guarda” e “prendi”. Utilizzando il primo comando viene visualizzata una descrizione della chiave inglese; usando il secondo comando l'avventuriero può portare con sé l'oggetto;
- *Un diario*: oggetto con cui è possibile interagire tramite il comando “guarda” e “leggi”. Utilizzando il primo comando viene visualizzata una descrizione esterna dell'oggetto; usando il secondo comando è possibile sfogliare una pagina alla volta e decidere se andare avanti o indietro nella lettura;
- *Un cartello*: oggetto con cui è possibile interagire tramite il comando “guarda”. Viene, quindi, visualizzata una scritta incisa su di esso.
- *Un computer*: oggetto con cui è possibile interagire tramite il comando “guarda”. Il protagonista può scegliere una tra le quattro opzioni disponibili:
 - 1) *Gestisci prenotazioni*: viene visualizzato un messaggio che avverte l'avventuriero che non è possibile accedere a questa sezione in quanto le prenotazioni possono essere gestite solo dal capo officina;
 - 2) *Visualizza persone in attesa*: viene visualizzata una sequenza di persone prenotate in ordine di arrivo;
 - 3) *Prenotati*: viene visualizzata una scritta che chiede all'utente di digitare il proprio nome, in modo tale che vada a finire in coda alle persone prenotate nella sezione *Visualizza persone in attesa*;
 - 4) *Spegni PC*: permette di spegnere il pc e tornare al gioco.

- *Un meccanico*: personaggio con il quale è possibile comunicare tramite il comando “parla”. Gli si può chiedere di cambiare la batteria dell’*automobile*, che si trova nel luogo “Stazione di servizio”, perdendo tuttavia del tempo messo a disposizione al giocatore. Questo personaggio può effettuare tale operazione un’unica volta e solo se l’auto si trova nel luogo “Meccanico” (con o senza l’avventuriero a bordo di essa). La sostituzione della batteria è essenziale per poter andare con l’automobile nelle città di “Bari”, “Roma” e “Pisa”, in quanto la vettura, dopo essere stata messa in moto utilizzando l’oggetto *chiave_auto*, che si trova nel luogo “Cabina secondo pilota”, manifesta segnali di usura della batteria.

3.15 CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”

Il progetto consiste nell’integrare un verbo “**ruba**” che farà rubare al giocatore alcuni oggetti presenti in determinate stanze del gioco. Sarà creato un inventario che verrà richiamato col verbo “**rubati**” che elencherà tutti gli oggetti rubati.

Verrà inserito un personaggio “**carabiniere**” che, per le prime 10 azioni del giocatore, resterà fisso in una stanza scelta, dopodiché comparirà randomicamente in una delle stanze del gioco dopo ogni azione del giocatore. Il “carabiniere”, quando incontrerà il giocatore, lo perquisirà controllando se in suo possesso avrà degli oggetti rubati: se non li trova il giocatore potrà continuare a giocare, altrimenti verrà portato in un luogo “**caserma**” creato appositamente dove dovrà restarci per un periodo di tempo, allo scadere del quale il giocatore potrà uscire dalla “caserma” e continuare a giocare.

Quando il “carabiniere” arresterà il giocatore, gli oggetti rubati saranno sequestrati. Col personaggio “carabiniere”, però, non sarà possibile interagirci.

3.16 ZAINO E VALIGIA

L’oggetto **Valigia** è stato pensato come una collezione di oggetti nella quale si ha una visione complessiva, mentre lo **zaino** è stato pensato come un contenitore di oggetti uno sopra l’altro, in cui è possibile prendere solo quello in cima.

Come già detto in fase di analisi, gli oggetti contenitori hanno dei limiti di peso trasportabile. La Valigia avrà dimensione n, mentre lo Zaino avrà dimensione m con $n > m$.

Se si possiede un oggetto contenitore, al momento della raccolta di un oggetto, il giocatore potrà decidere se portare con sé l’oggetto o metterlo nel contenitore, perciò verrà posta la seguente domanda:

“Se vuoi portare l’oggetto con te, premi (y);
se vuoi metterlo in valigia, premi (v)”

in caso si sia raccolta la valigia oppure:

“Se vuoi portare l’oggetto con te, premi (y);
se vuoi metterlo nello zaino, premi (z)”

in caso si sia raccolto lo zaino.

In base alla risposta, l’oggetto verrà messo o nell’inventario o nel contenitore.

L’oggetto **Valigia** verrà realizzato mediante la struttura di dati insieme, in quanto l’oggetto è stato pensato come una collezione di oggetti.

La struttura di dati insieme intende riprodurre la nozione matematica di insieme, ossia una collezione (o famiglia) di elementi (detti membri) tutti appartenenti ad uno stesso tipo base (detto dominio).

Ogni stanza d'insieme sarà un elemento dell'insieme delle parti del dominio.

È una struttura omogenea, non-lineare, dinamica, ad accesso diretto e in memoria centrale. A differenza delle liste gli elementi non sono caratterizzati da una posizione, e dunque non possono apparire più di una volta.

Il numero di elementi di un insieme i (detto cardinalità e denotato da $|i|$) rappresenta la dimensione dell'insieme.

Soltanente rappresentati graficamente tramite diagrammi di Venn, in matematica gli insiemi possono essere definiti:

1. estensionalmente, cioè elencandone tutti i membri,
2. intenzionalmente cioè specificando la proprietà caratteristica in base alla quale stabilire se un qualunque dato elemento del dominio è membro dell'insieme o meno.

In informatica ci si riferisce al modo estensionale.

La relazione fondamentale è quella di appartenenza di un elemento x ad un insieme i ($x \in i$), in base alla quale e poi definita la relazione di inclusione di un insieme i' in un insieme i'' ($i' \subseteq i''$).

Le operazioni principali fra due insiemi, i' e i'' sono unione ($i' \cup i''$), intersezione ($i' \cap i''$) e differenza ($i' \setminus i''$), ben note in matematica. Un insieme che non ha elementi è detto vuoto e viene indicato con \emptyset .

L'oggetto **Zaino** verrà creato mediante la struttura di dati pila.

Una pila è una sequenza di elementi omogenei, dove è possibile aggiungere o cancellare elementi da un estremo (la testa).

È una struttura lineare e dinamica.

Può essere vista come un caso speciale di lista in cui l'ultimo elemento inserito è il primo ad essere rimosso (LIFO). Tale accesso è diretto.

4 REALIZZAZIONE

4.1 MUSEO

Per realizzare questo sistema sono stati aggiunti i seguenti file:

- Museo.h
- Museo.cpp
- Grafo.h
- Adiacenza.h
- Lista.h
- Cella_L_DP.h
- StanzaMuseo.h
- StanzaMuseo.cpp

Grafo.h è la realizzazione del grafo. È stata scelta la realizzazione “Vettore con liste di adiacenza” considerando che, per la sua natura, l’operatore che restituisce la lista di adiacenza ha complessità ridotta e verrà utilizzato per creare l’elenco di movimenti disponibili per l’utente.

Adiacenza.h è la realizzazione di ogni elemento nella lista di adiacenti di ogni nodo, dotata di un riferimento al nodo a cui è adiacente e di un campo per un possibile peso dell’arco nel caso in cui esista.

Lista.h e Cella_L_DP.h sono le realizzazioni necessarie al funzionamento della lista, utilizzata per le liste di adiacenza. È stata scelta la realizzazione con doppi puntatori in modo tale da legare in maniera efficiente ed efficace il vettore di nodi del grafo alle liste di adiacenza.

Per realizzare il museo è stata scritta una nuova classe nel file “Museo.h” (la sua implementazione in Museo.cpp), che sfrutta il grafo e i suoi operatori per gestire le stanze e gli spostamenti.

Operatori della classe Museo:

- creamuseo () → Museo
- getstanzacorrente () → Nodo
- setstanzacorrente(Nodo) → Museo
- getstanzeadiacenti() → Lista
- getiniziale() → Nodo
- leggiStanza(Nodo) → StanzaMuseo
- scriviStanza (Nodo, StanzMuseo) → ()

Variabili della classe Museo:

- stanzainiziale , di tipo Nodo (utilizzata per tenere traccia della stanza di entrata del museo);
- stanzacorrente, di tipo Nodo (utilizzata per tenere traccia della stanza nella quale l’utente si trova);
- grafomuseo, di tipo Grafo (utilizzata per contenere la mappa del museo con relative stanze);
- titoli, array di tipo stringa (utilizzato per memorizzare i titoli delle stanze);
- descrizioni, array di tipo stringa (utilizzato per memorizzare le descrizioni delle stanze).

StanzaMuseo.h contiene la classe che rappresenterà le stanze del museo come campo informativo nei nodi del grafo, in StanzaMuseo.cpp è presente la sua implementazione.

Operatori della classe StanzaMuseo:

- setdescrizione(string) → StanzaMuseo

- `getdescrizione() → string`
- `settitolo(string) → StanzaMuseo`
- `gettитоlо() → string`

Variabili della classe StanzaMuseo:

- `descrizione`, di tipo stringa (utilizzata per la descrizione della stanza del museo);
- `titolo`, di tipo stringa (utilizzata per il titolo della stanza del museo).

Inoltre per realizzare il sistema sono stati modificati i file:

- `Astro.cpp`, precisamente l'azione 96
- `Gioco.cpp`, commentando i metodi ormai superflui.

4.2 SCUOLA E AULE

I file che sono stati modificati sono:

- `Astro.h`
- `Gioco.h`
- `Astro.cpp`
- `Mappa.nav`

File aggiunti nella cartella descrizioni:

- `28.txt`
- `29.txt`
- `30.txt`
- `31.txt`

4.2.1 Scuola e le classi 1A, 2A, 3A

Ai nuovi luoghi da implementare sono stati assegnati delle caratteristiche univoche che per facilitare la lettura e la comprensione e sono stati inseriti all'interno da una tabella. Le caratteristiche sono:

- **Etichetta:** stringa che identifica il nome del luogo;
- **Codice Luogo:** stringa che identifica i percorsi utilizzabili all'interno del luogo. Tale codice è formato da 12 caratteri aventi il seguente significato NNSSEEOOUUDD (N = Nord, S = Sud, E = est, O = ovest, U = Salì, D = Scendi);
- **Codice Mappa:** codice numerico univoco che lo contraddistingue dagli altri luoghi presenti nel gioco.

ETICHETTA	CODICE LUOGO	CODICE MAPPA
Scuola	300129310000	28
1A	000000280000	29
2A	002800000000	30
3A	000028000000	31

Dal codice luogo è possibile notare che la Scuola è stato inserita a Nord della cabina di pilotaggio (il codice Mappa della cabina di pilotaggio è 1). La classe 1A è stata inserita a est dalla Scuola, la 2A a nord della Scuola e la 3A a ovest dalla Scuola.

4.2.2 Oggetti

A ciascun oggetto da implementare sono state assegnate univoche, e sono:

- **Etichetta:** stringa che identifica il nome dell'oggetto

- **Vocabolo:** vocabolo accettato all'interno del gioco, parola chiave da scrivere per poter interagire con quell'oggetto.
- **Codice Oggetto:** codice univoco con cui è rappresentato quel particolare oggetto all'interno del gioco.
- **Codice Mappa:** codice univoco della stanza in cui è presente quel particolare oggetto. Il segno “-”, indica che l'oggetto non può essere trasportato.

Rappresentiamo in una tabella i seguenti oggetti (gli oggetti banchi, sedie, cattedra saranno presenti in ogni classe):

ETICHETTA	VOCABOLO	CODICE OGGETTO	CODICE MAPPA
una cartina galattica	cartina	60	-28
il Preside	preside	73	-28
la maestra Clara	maestra	73	-29
una cartina geografica	cartina	60	-29
dei banchi	banchi	43	-29
delle sedie	sedie	56	-29
una cattedra	cattedra	70	-29
la maestra Mara	maestra	73	-30
dei banchi	banchi	43	-30
delle sedie	sedie	56	-30
una cattedra	cattedra	70	-30
un registro	registro	88	-30
dei banchi	banchi	43	-31
una cattedra	cattedra	70	-31
delle sedie	sedie	56	-31
una lavagna	lavagna	90	-31

4.2.3 Azioni

A ciascuna azione da integrare sono state assegnate delle caratteristiche univoche. Le caratteristiche sono:

- **Comando:** comanda che il giocatore intende far eseguire al protagonista dell'avventura.
- **Codifica:** codice numerico ottenuto tramite l'espressione:
 - [Codifica comando= LL*10000+VV*100+OO]
 - LL=Codice del luogo in cui il comando può essere impartito.
 - VV=Codice del verbo che indica una certa azione.
 - OO=Codice dell'oggetto.
- **Codice Oggetto:** codice oggetto con cui è rappresentato quel comando all'interno del gioco.

COMANDO	CODIFICA	CODICE OGGETTO
leggi registro	2588	97
guarda registro	1088	97
guarda lavagna	311090	98
parla con il preside	283973	100
parla con la maestra Clara	293973	101
parla con la maestra Mara	303973	102
guarda cartina galattica	291060	103
guarda cartina geografica	291060	104

4.3 ARCHIVIO E MUSEO (VECCHIA VERSIONE DEL MUSEO)

Vediamo com'è stato possibile realizzare il tutto, e quali file sono stati modificati.

Le modifiche effettuate fanno riferimento ai seguenti file:

- Astro.h
- Gioco.h
- Astro.cpp
- Gioco.cpp
- Mappa.nav

File aggiunti:

- 26.txt e 27.txt (Cartella Descrizioni)

Per poter integrare correttamente le parti mancanti nel progetto base sono state apportate le modifiche necessarie, in modo da non creare conflitti con gli oggetti e i luoghi già presenti.

4.3.1 Archivio e Museo

Ai nuovi luoghi da integrare sono stati assegnati delle caratteristiche univoche che, per facilitarne la lettura e comprensione, sono state racchiuse nella seguente tabella:

- **Etichetta:** stringa che identifica il nome del luogo.
- **Codice Luogo:** codice numerico che identifica i percorsi utilizzabili all'interno del luogo. Tale codice è formato da 12 cifre aventi il seguente significato: NNSSEEOOUUDD (N=Nord, S=Sud, E=Est, O=Ovest, U=Sali, D=Scendi).
- **Codice Mappa:** codice numerico univoco che lo contraddistingue dagli altri luoghi presenti nel gioco.

ETICHETTA	CODICE LUOGO	CODICE MAPPA
Archivio	252700000000	26
Museo	260000000000	27

Dal Codice Luogo è dunque possibile notare che L'Archivio è stata inserito a Sud dell'Aula (il cui codice mappa è 25). Mentre il Museo a Sud dell'Archivio. Dunque dal Museo si potrà andare solo verso Nord in Archivio, mentre dall'Archivio si potrà andare a Sud nel Museo oppure tornare a Nord nell'Aula.

4.3.2 Oggetti

A ciascun oggetto da integrare sono state assegnate delle caratteristiche univoche che, per facilitarne la lettura e comprensione, sono state racchiuse nella seguente tabella:

- **Etichetta:** stringa che identifica il nome dell'oggetto.
- **Vocabolo:** vocabolo accettato all'interno del gioco, parola chiave da scrivere per poter interagire con quell'oggetto.
- **Codice Oggetto:** codice univoco con cui è rappresentato quel particolare oggetto all'interno del gioco.
- **Codice Mappa:** codice univoco della stanza in cui è presente quel particolare oggetto. Il segno '-' indica che l'oggetto non può essere trasportato.

ETICHETTA	VOCABOLO	CODICE OGGETTO	CODICE MAPPA
Il File12	file12	37	-26
Il File13	file13	57	-26
Il File15	file15	58	-26
Il File21	file21	71	-26
Un Computer	computer	99	-26

4.3.3 Azioni

A ciascuna azione da integrare sono state assegnate delle caratteristiche univoche che, per facilitarne la lettura e comprensione, sono state racchiuse nelle seguenti tabelle:

- **Comando:** comando che il giocatore intende far eseguire al protagonista dell'avventura.
- **Codifica:** codice numerico univoco ottenuto tramite l'espressione:
 - [Codifica comando= LL*10000+VV*100+OO]
 - LL=Codice del luogo in cui il comando può essere impartito.
 - VV=Codice del verbo che indica una certa azione.
 - OO=Codice dell'oggetto.
- **Codice Oggetto:** codice univoco con cui è rappresentato quel comando all'interno del gioco.

COMANDO	CODIFICA	CODICE OGGETTO
Usa computer	262999	91
Leggi file12	262537	92
Leggi file13	262557	93
Leggi file15	262558	94
Leggi file21	262571	95
Entra nel (museo)	277416	96

4.4 AULA SINGOLA

Vedremo adesso le realizzazioni delle scelte progettuali e delle decisioni prese durante la fase precedente.

Per poter integrare correttamente le parti mancanti nel progetto base, sono stati controllati tutti i codici del progetto di Schirano Giuseppe, apportando le modifiche necessarie, in modo da non creare conflitti con gli oggetti già presenti in Galeandro Germano.

4.4.1 Aula

Al nuovo luogo da integrare sono state assegnate delle caratteristiche univoche che, per facilitarne la lettura e comprensione, sono state racchiuse nella seguente tabella:

- Etichetta: stringa che identifica il nome del luogo.
- Codice Luogo: codice numerico che identifica i percorsi utilizzabili all'interno del luogo. Tale codice è formato da 12 cifre aventi il seguente significato: NNSSEEOOUUDD (N=Nord, S=Sud, E=Est, O=Ovest, U=Sali, D=Scendi).
- Codice Mappa: codice numerico univoco che lo contraddistingue dagli altri luoghi presenti nel gioco.

ETICHETTA	CODICE LUOGO	CODICE MAPPA
Aula	040000000000	25

Dal Codice Luogo è dunque possibile notare che l'Aula è stata inserita a Sud dell'estremità sud del corridoio. Per cui, una volta entrati nell'aula, sarà solo possibile andare solo a Nord, tornando così nell'estremità sud del corridoio (il cui codice mappa è 04).

4.4.2 Oggetti

A ciascun oggetto da integrare sono state assegnate delle caratteristiche univoche che, per facilitarne la lettura e comprensione, sono state racchiuse nella seguente tabella:

- Etichetta: stringa che identifica il nome dell'oggetto.
- Vocabolo: vocabolo accettato all'interno del gioco, parola chiave da scrivere per poter interagire con quell'oggetto.

- Codice Oggetto: codice univoco con cui è rappresentato quel particolare oggetto all'interno del gioco.
- Codice Mappa: codice univoco della stanza in cui è presente quel particolare oggetto. Il segno “-“ indica che l'oggetto non può essere trasportato.

ETICHETTA	VOCABOLO	CODICE OGGETTO	CODICE MAPPA
un computer	computer	78	-25
delle sedie	sedia	92	25
una lezione in corso	lezione	71	-25
un proiettore	proiettore	80	-25
dei libri	libri	55	25

4.4.3 Azioni

A ciascuna azione da integrare sono state assegnate delle caratteristiche univoche che, per facilitarne la lettura e comprensione, sono state racchiuse nelle seguenti tabelle:

- Azione: azione che si intende eseguire.
- Vocabolo: vocabolo accettato all'interno del gioco, parola chiave da scrivere per poter eseguire quell'azione.
- Codice Oggetto: codice univoco con cui è rappresentata quell'azione all'interno del gioco.

AZIONE	VOCABOLO	CODICE OGGETTO
Accendere (il proiettore)	accendi	86
Spegnere (il proiettore)	spegni	72

Queste azioni potranno essere richiamate in qualsiasi momento, seguite dal vocabolo “proiettore”.

AZIONE	VOCABOLO	CODICE OGGETTO
Seguire (una lezione)	segui	10

Questa azione potrà essere richiamata in qualsiasi momento, seguita dal vocabolo “lezione”.

AZIONE	VOCABOLO	CODICE OGGETTO
Sedersi (sulla sedia)	siediti	94

Questa azione potrà essere richiamata in qualsiasi momento, seguita dal vocabolo “sedia”, oppure da nessun vocabolo.

AZIONE	VOCABOLO	CODICE OGGETTO
Usa (computer)	usa	29

Questa azione potrà essere richiamata in qualsiasi momento, seguita dal vocabolo “computer”.

AZIONE	VOCABOLO	CODICE OGGETTO
Leggi (libri)	leggi	25

Questa azione potrà essere richiamata in qualsiasi momento, seguita dal vocabolo “computer”.

NOTA: I vocaboli “siediti”, “usa” e “leggi” non sono stati implementati in quanto già presenti nel gioco, nella sua versione precedente.

4.4.4 Comandi

A ciascun comando da integrare sono state assegnate delle caratteristiche univoche che, per facilitarne la lettura e comprensione, sono state racchiuse nella seguente tabella:

- Comando: comando che il giocatore intende far eseguire al protagonista dell'avventura.
- Codifica: codice numerico univoco ottenuto tramite l'espressione:
 - [Codifica comando= LL*10000+VV*100+OO]
 - LL=Codice del luogo in cui il comando può essere impartito.

- VV=Codice del verbo che indica una certa azione.
- OO=Codice dell'oggetto.
- Codice Oggetto: codice univoco con cui è rappresentato quel comando all'interno del gioco.

COMANDO	CODIFICA	CODICE OGGETTO
siediti	259400	85
siediti sedia	259492	85
leggi libri	252555	86
usa computer	252978	87
segui lezione	251071	88
accendi proiettore	258680	89
spegni proiettore	257280	90

4.5 UFFICIO POSTALE

Per l'integrazione nell'adventure sarà inoltre necessario l'inserimento dell'Ufficio Postale nella Mappa e la creazione di un file 24.txt per la descrizione del luogo al momento dell'accesso.

4.5.1 Mappa.nav

Codice modificato :

- Riga 1 : 23 -> 24

Nuove righe di codice :

- 12, Nell'ufficio/deposito, 002414000200, via 1,1,1, via 6,2,2
- 24, Nell'Ufficio Postale, 120000000000, via 3,2,2
- 12,24, via 1, sud,6,8,1,1
- 24,12, via 1, nord,6,8,1,1

4.5.2 24.txt

Nell'Ufficio Postale

Nell'Ufficio Postale

Di nuovo nell'Ufficio Postale

Nuovamente nell'Ufficio Postale

Ancora una volta nell'Ufficio Postale

4.5.3 Inserimento oggetto "Sportello Telematico"

L'oggetto Sportello Telematico sarà presente nell'Ufficio Postale (Luogo 24). Sarà visibile ma non trasportabile e sarà possibile interagire con esso tramire il comando "guarda".

Prima di interagire con il sistema vero e proprio l'utente dovrà decidere se attendere o meno il proprio turno in fila.

Verrà mostrato a video un messaggio che lo avverrà del numero di persone già presenti in fila e gli verrà posta la domanda : "Ci sono X persone in fila, vuoi attendere?" (dove X è un numero pseudo-randomico compreso tra 0 e 10).

Se l'utente digiterà "s" avrà accesso allo Sportello Telematico e verrà scalato il tempo a disposizione in base al numero di persone presenti in fila : tempo = tempo – p.

Se l'utente digiterà "n" non avrà accesso allo Sportello Telematico ma il tempo a disposizione verrà scalato solo di un'unità come impostazione standard dell'Adventure.

L'oggetto sarà identificato dall'oggetto "uno sportello telematico" associato al vocabolo "sportello" (99)

Accedendo allo Sportello Telematico si avranno le seguenti interazioni possibili

- Accedere alla sezione di invio (tramite il comando "1")
- Accedere alla sezione di ricezione (tramite il comando "2")
- Uscire dallo Sportello telematico (tramite il comando "0")

Dall'area di invio sarà possibile :

- Inviare una richiesta di suggerimento (tramite il comando "1")
 - Il suggerimento verrà inviato tramite una lettera che sarà disponibile nella sezione di ricezione dopo la richiesta. Sarà possibile ricevere solo un suggerimento
- Inviare una richiesta per ottenere un oggetto misterioso (tramite il comando "2")
 - L'oggetto sarà accessibile dopo aver aperto un pacco disponibile nella sezione di ricezione dopo la richiesta. Sarà possibile ricevere un solo oggetto misterioso.
- Inviare un oggetto posseduto dal giocatore a dei luoghi predefiniti dell'astronave (tramite il comando "3")
 - L'azione sarà possibile ovviamente se si avranno oggetti disponibili nell'inventario
 - I luoghi disponibili per l'invio sono la Sala del Reattore (tramite comando "8") e la Cabina del Secondo Pilota (tramite comando "5")
- Depositare del denaro in Banca (tramite il comando "4")
 - L'azione sarà possibile se si possiederà l'oggetto Portafoglio necessario per contenere i soldi.
 - Tutti i soldi presenti nel Portafoglio verranno poi resi disponibili nel conto in Banca del giocatore.

Sarà possibile accedere all'area di ricezione solo se in possesso dell'oggetto "Documento d'identità" e si avrà accesso alle seguenti funzioni :

- Ricevere una lettera (dopo la richiesta nell'area di invio)
 - La lettera verrà automaticamente trasferita nell'inventario una volta entrati nel sistema di ricezione
- Ricevere un pacco (dopo la richiesta nell'area di invio)
 - Il pacco verrà automaticamente trasferito nell'inventario una volta entrati nel sistema di ricezione e conterrà l'oggetto misterioso (la chiave del secondo pilota)

4.5.4 Inserimento oggetto "Documento d'Identità"

Il Documento d'Identità sarà un oggetto visibile e prendibile. Sarà necessario al fine di accedere al sistema di ricezione dello Sportello Telematico e la sua posizione originaria sarà nella "Tua cabina di Pilotaggio" (luogo 6).

Sarà identificato dall'oggetto "un documento d'identità" associato al vocabolo "documento" (62)

Le interazioni possibili con il Documento d'identità saranno :

- Prendi documento
- Lascia documento

4.5.5 Inserimento oggetto "Lettera"

La lettera sarà un oggetto non visibile (Luogo -99) ma prendibile. Per ottenere la lettera sarà necessario fare la richiesta allo Sportello telematico nel sistema di invio per poi riceverla nel sistema di ricezione.

Sarà possibile ottenere una sola lettera di suggerimento.

Sarà identificato dall'oggetto "una lettera" associato al vocabolo "lettera" (37)

Le interazioni possibili con la Lettera saranno :

- Leggi lettera
- Lascia lettera
- Prendi lettera

Una volta aperta la lettera verrà visualizzato il seguente messaggio :

"Hai aperto la lettera"

"Messaggio : La chiave dell'armadietto del secondo pilota potrebbe essere andata perduta! . . .

Per fortuna che alla base hanno sempre i pezzi di ricambio!"

4.5.6 Inserimento oggetto "Pacco"

Il Pacco sarà un oggetto non visibile (Luogo -99) ma prendibile. Per ottenere il Pacco sarà necessario fare una richiesta di invio (dell'oggetto misterioso) allo Sportello Telematico e poi riceverlo tramite il sistema di ricezione.

Sarà possibile ottenere un solo Pacco e conterrà sempre la Chiave del secondo pilota.

Sarà identificato dall'oggetto "un pacco" associato la vocabolo "pacco" (57)

Le interazioni possibili con il Pacco saranno :

- Apri pacco
- Lascia pacco
- Prendi pacco

Una volta aperto il pacco non sarà più presente nell'inventario del giocatore e al suo posto sarà presente la Chiave del secondo pilota.

4.6 DIALOGHI

Vedremo adesso come è stato possibile realizzare il tutto e quali file sono stati modificati.

Le modifiche effettuate fanno riferimento ai seguenti file:

- Personaggi.h
- Personaggi.cpp
- Gioco.h
- Gioco.cpp

I file aggiunti al progetto per l'implementazione dei dialoghi sono:

- AlberoNario.h
- Binalbero.h
- Nodo_Albero_Binario.h
- Domanda.h
- Domanda.cpp

- Dialogo.h
- Dialogo.cpp
- Dialoghi.txt
- Statidialoghi.txt
- AstroStatiDialoghi.txt (viene creato automaticamente quando si salva)

4.7 SALA SCOMMESSE E SIMULATORE

4.7.1 Aggiunta oggetti e vocaboli nel gioco

Sono stati aggiunti all'interno del file Astro.cpp il nuovo oggetto previsto in fase di progettazione ed i rispettivi vocaboli.

4.7.1.1 *Vocaboli*

```
vocabolario.inserisci("simulatore",58);
vocabolario.inserisci("avvia",14);
vocabolario.inserisci("gioca",44);
```

4.7.1.2 *Azioni*

```
azioni.inserisci(234400, 80); //azione per giocare nella stanza sala scommesse
azioni.inserisci(232958, 80); //azione sinonimo usa simulatore
azioni.inserisci(231058, 81); //azione per guardare il simulatore
```

4.7.1.3 *Oggetti*

```
oggetti.inserisci(Oggetto("un simulatore",58, -23));
```

4.8 AUDITORIUM

Per aggiungere il luogo auditorium sono stati modificati i seguenti file:

- Astro.h
- Gioco.h
- Gioco.cpp
- Astro.cpp
- Mappa.nav

È stato aggiunto un nuovo file nella cartella “Descrizioni” per il nuovo luogo, di nome 32.txt .
Inoltre sono state create le seguenti classi:

- Canzone.h
- Canzone.cpp
- Luce.h
- Luce.cpp

4.8.1 Classi create

Canzone: La classe “Canzone” contiene le informazioni riguardanti un determinato brano musicale. Le informazioni sono il nome del brano, l'artista e l'anno di registrazione. Gli operatori utilizzati sono:

- Canzone() ---> Canzone ///Costruttore

- `get_nome()`---> stringa
- `get_artista` ----> stringa
- `get_anno` ----> intero
- `set_nome(stringa)` ----> Canzone
- `set_artista(stringa)`----> Canzone
- `set_anno(stringa)`----> Canzone
- Overload Operatore << ----> stampa un messaggio con le informazioni del brano, senza dover usare gli operatori get nel programma.
- Variabili: nome (stringa), artista (stringa), anno (intero)

Luce: La classe luce serve per rappresentare l'entità luce. Attualmente è composta solamente da un attributo, il nome, che indica la posizione nella stanza (es. "destra avanti"). La classe è stata creata per facilitare un'eventuale espansione futura del concetto di luci nell'astronave. Gli operatori usati sono:

- `Luce()` ----> Luce //Costruttore
- `getNome()` ----> stringa
- `setNome(stringa)` ----> Luce
- Overload operatore << ---> stampa un messaggio di info della luce
- Overload operatore ==
- Overload operatore !=
- Overload operatore <
- Overload operatore <=
- variabili: nomeLuce (stringa)

4.8.2 Realizzazioni usate

È stata utilizzata una **lista con doppi puntatori (collegata)** per il Jukebox poiché offre complessità costante anche quando bisogna tornare all'elemento precedente (operazione che assume complessità lineare nel caso di una lista senza il puntatore che punta all'elemento precedente).

Per la playlist è stata utilizzata una **coda con puntatori**, in maniera tale da rendere di complessità costante tutti gli operatori della lista.

Per l'insieme, invece, è stata utilizzata una realizzazione con **lista non ordinata** poiché il dominio non è molto esteso (le luci accese nell'auditorium), quindi non otterremo vantaggio concreto utilizzando una lista ordinata.

4.8.3 Modifica Mappa

Per implementare l'auditorium, accessibile scendendo dal luogo "Corridoio" sono state assegnate delle informazioni univoche, che sono l'etichetta (stringa che identifica il luogo), il codice del luogo (stringa che identifica quali sono i percorsi utilizzabili dentro quel determinato luogo) e il codice della mappa (codice univoco per contraddistinguerlo dagli altri luoghi del gioco).

L'auditorium ha:

- **Etichetta:** Auditorium
- **Codice Luogo:** 000000000300
- **Codice Mappa:** 32

Inoltre è stato necessario modificare il codice luogo del Corridoio (si trova ad est dal punto di inizio del gioco) per rendere accessibile l'auditorium.

- **Vecchia etichetta:** 0204120619000

- **Nuova etichetta:** 0204120619332

Inoltre sono state aggiunte le righe “3,32 via 1,scendi,2,2,1,1” e “32,3 via 1,sali,2,2,1,1” per poter creare la strada dal corridoio all'auditorium.

4.8.4 Oggetti

Sono stati implementati nuovi oggetti e vocaboli per aumentare l'interattività nell'Auditorium.

Ogni oggetto ha delle informazioni univoche che sono l'**etichetta** (identifica il nome dell'oggetto), il **vocabolo** (per far comprendere all'interprete del gioco la parola chiave con cui si può interagire con l'oggetto), il **codice oggetto** (rappresenta quel determinato oggetto; deve essere univoco se l'oggetto deve essere prenibile dal giocatore) e il **codice mappa** (identifica il luogo in cui si trova quel determinato oggetto).

Di seguito si riportano tutti i nuovi oggetti inseriti:

ETICHETTA	VOCABOLO	CODICE OGGETTO	CODICE MAPPA
Un proiettore rotto	Proiettore rotto	80	-32
Uno schermo	Schermo	60	-32
Una finestra grande	Finestra	90	-32
Un microfono	Microfono	90	-32
Dei strumenti musicali	Strumenti Strumento	88	-32
Una scacchiera sulla cattedra	Scacchiera	73	-32
Un jukebox antico	Jukebox	56	-32
Un pannello di controllo delle luci	Pannello	70	-32

4.8.5 Azioni

Per rendere interagibile ogni oggetto inserito nell'Auditorium sono state implementate nuove azioni. Un'azione ha un **comando** (il comando che serve digitare per interagire con un oggetto specifico), una **codifica** (e' un codice numerico ottenuto tramite un'espressione matematica

[Codice_Luogo * 10000 + Codice_Verbo * 100 + Codice_Oggetto]

che permette di avere sempre una codifica univoca) e un **codice azione** (che identifica l'azione da effettuare).

Di seguito si riportano le nuove azioni inserite:

COMANDO	CODIFICA	CODICE AZIONE
Accendi proiettore	328680	105
Guarda schermo	321060	106
Guarda finestra	321090	107
Parla microfono	323990	108
Usa Jukebox	322956	109
Guarda strumenti/o	321088	110
Suona strumenti	327288	111

Usa pannello	322970	112
Guarda scacchiera	321073	113

4.9 STAZIONE

Il luogo “Stazione” è stato inserito a nord dell’ufficio/deposito, infatti partendo da “La mia cabina”, bisognerà passare per il corridoio (andando ad “Est”), per l’estremità nord del corridoio (andando a “Nord”) e per l’ufficio/deposito (con il comando “scendi”).

Per effettuare l’integrazione della “Stazione”, a tale luogo sono state assegnate le seguenti caratteristiche univoco:

- **CODICE DEL LUOGO:** codice numerico che identifica il luogo
- **NOME DEL LUOGO:** stringa contenente il nome del luogo
- **CODICE DELLE DIREZIONI:** codice che consente di sapere le direzioni in cui è possibile andare (partendo dall’interno del luogo). Il codice è nel formato: NNSSEOOOUUDD, ovvero “Nord”, “Sud”, “Est”, “Ovest”, “Sali”, “Scendi”.

CODICE LUOGO	NOME LUOGO	CODICE DIREZIONI
33	Stazione	001200000000

In questo luogo saranno presenti quattro oggetti: un treno, una biglietteria, una panchina ed un vecchio giornale.

- **Panchina:** l’oggetto chiamato “panchina” è un oggetto non trasportabile. Il giocatore potrà sedersi sulla panchina o alzarsi dalla panchina, ma non sarà utile allo svolgimento del gioco.
- **Treno:** l’oggetto chiamato “treno” è un oggetto non trasportabile. Il giocatore, una volta acquistato il biglietto, potrà salire sul treno. Però, a causa di guasto, il treno non può partire e quindi il giocatore avrà inutilmente speso tempo e denaro.
- **Biglietteria:** l’oggetto chiamato “biglietteria” è un oggetto non trasportabile. Il giocatore potrà utilizzare la biglietteria per acquistare i biglietti del treno. L’utente potrà pagare in contanti, quindi per poter usufruire dei servizi offerti dalla biglietteria deve assolutamente essere in possesso del portafoglio (con relativi contanti). Inoltre, per assumere un aspetto più realistico, ci sarà una coda di persone in attesa del proprio turno.
- **Vecchio giornale:** l’oggetto chiamato “giornale” è un oggetto trasportabile. Anche questo oggetto non rappresenta un’utilità per la soluzione del gioco.

Ad ogni oggetto sono associate le seguenti caratteristiche:

- **CODICE DELL’OGGETTO:** codice (univoco) numerico che identifica l’oggetto.
- **NOME DELL’OGGETTO:** stringa contenente il nome dell’oggetto.
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l’oggetto all’interno del gioco.
- **CODICE MAPPA:** codice univoco che identifica la stanza in cui è presente l’oggetto. Il simbolo “-“ davanti al codice della mappa indica che quell’oggetto non è trasportabile.

CODICE OGGETTO	NOME OGGETTO	VOCABOLO	CODICE MAPPA
80	Una biglietteria	biglietteria	-33
78	Un giornale strappato	giornale	33
55	Un treno	treno	-33
71	Una panchina	panchina	-33

Inoltre, sono state implementate le preposizioni “sul, sulla” per poter effettuare le azioni (sali “sul” treno, siediti “sulla” panchina) che verranno descritte in seguito.

VOCABOLO	CODICE OGGETTO
----------	----------------

Sul	7
Sulla	7

Ad ogni azione sono state assegnate le seguenti caratteristiche univoche:

- **CODICE DELL'OGGETTO:** codice (univoco) che identifica il verbo
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l'oggetto all'interno del gioco.
- **AZIONE:** azione che si intende eseguire

CODICE OGGETTO	VOCABOLO	AZIONE
94	Siediti	Sedersi sulla panchina
29	Usa	Usa biglietteria
25	Leggi	Leggi giornale
5	Sali	Sali sul treno

Per ogni comando sono state assegnate le seguenti caratteristiche univoche:

- **CODICE DELL'OGGETTO:** codice (univoco) che identifica il comando
- **CODIFICA:** codice (univoco) nel formato LLVVOO, ricavato dall'espressione LL*10000 + VV*100 + OO.
- **COMANDO:** consentirà l'esecuzione dell'azione

CODICE OGGETTO	CODIFICA	COMANDO
114	339471	Siediti (sulla) panchina
115	332980	Usa biglietteria
116	332578	Leggi giornale
117	330555	Sali (sul) treno

4.10 PALESTRA

Il luogo “Palestra” è stato inserito scendendo dalla “Cabina del secondo pilota”, infatti partendo da “La mia cabina”, bisognerà andare passare per il corridoio (andando ad “Est”), per l'estremità nord del corridoio (andando a “Nord”) successivamente per la cabina per il secondo pilota (andando a “Ovest”) e poi infine scendere per la Palestra (“scendi”)

La sequenza di comandi per accedere al luogo Palestra è: est, nord, ovest, scendi.

Per effettuare l'integrazione della “Palestra”, a tale luogo sono state assegnate le seguenti caratteristiche univoche.

- **CODICE DEL LUOGO:** codice numerico che identifica il luogo
- **NOME DEL LUOGO:** stringa contenente il nome del luogo
- **CODICE DELLE DIREZIONI:** codice che consente di sapere le direzioni in cui è possibile andare (partendo dall'interno del luogo). Il codice è nel formato: NNSSEEOOUUDD, ovvero “Nord”, “Sud”, “Est”, “Ovest”, “Sali”, “Scendi”.

CODICE DEL LUOGO	NOME DEL LUOGO	CODICE DIREZIONI
34	Palestra	000000000500

Come già accennato nel capitolo relativo all'analisi dell'integrazione, in questo luogo saranno presenti 4 oggetti: una panca, un tapis roulant, un terminale informativo, e delle schede.

4.10.1 Oggetti

- **Panca:** l'oggetto chiamato “panca” è un oggetto non trasportabile. Il giocatore potrà usare la panca e decidere quante ripetizioni fare, incrementerà i valori di “costituzione”. Ogni ripetizione vale ad un 1 secondo di tempo reale, ma che aumenta del 20% per ogni ripetizione effettuata nella stessa serie.

- **Tapis roulant:** l'oggetto chiamato “tapis/roulant” è un oggetto non trasportabile. Il giocatore potrà usare la panca, e scegliere per quanti minuti correre, il valore di resistenza aumenterà, ma bisogna tenere conto che il tempo per finire il gioco diminuirà. Un minuto sul tapis roulant equivale a 3 secondi di tempo reale
- **Terminale informativo:** l'oggetto chiamato “terminale” è un oggetto non trasportabile. Il giocatore una volta accesso il terminale potrà capire il funzionamento del luogo palestra, i suoi oggetti, e le statistiche andranno modificate.
- **Scheda:** l'oggetto chiamato “scheda”, è un oggetto non trasportabile. Il giocatore potrà creare o usare schede precedentemente create, ciascuna permetterà di eseguire una sequenza fissata di esercizi, e al momento della creazione possiamo stabilire arbitrariamente il numero di esercizi sugli oggetti di “panca” o “tapis” in maniera libera, e stabilire per ognuna il numero di ripetizioni, o i minuti di corsa.

Ad ogni oggetto sono state associate le seguenti caratteristiche:

- **CODICE DELL'OGGETTO:** codice (univoco) numerico che identifica l'oggetto.
- **NOME DELL'OGGETTO:** stringa contenente il nome dell'oggetto
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l'oggetto all'interno del gioco
- **CODICE MAPPA:** codice univoco che identifica la stanza in cui è presente l'oggetto. Il simbolo “-“ davanti al codice della mappa indica che quell'oggetto non è trasportabile

CODICE OGGETTO	NOME OGGETTO	VOCABOLO	CODICE MAPPA
57	Una panca	panca	-34
47	Un tapis roulant	tapis/roulant	-34
99	Terminale informativo	terminale	-34
84	Schede di allenamento	scheda/schede	-34

4.10.2 Azioni

Ad ogni azione sono state assegnate le seguenti caratteristiche univoche:

- **CODICE DELL'OGGETTO:** codice (univoco) che identifica il verbo.
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l'oggetto all'interno del gioco.
- **AZIONE:** l'azione che si intende eseguire.

CODICE OGGETTO	VOCABOLO	AZIONE
29	Usa	Allenarsi con la panca
29	Usa	Allenarsi con il tapis roulant
29	Usa	Guardare il terminale
10	Guarda	Guardare il terminale
29	Usa	Visualizzare o creare le schede

4.10.3 Comandi

Per ogni comando sono state assegnate le seguenti caratteristiche univoche:

- **CODICE AZIONE:** codice (univoco) che identifica il comando
- **CODIFICA:** codice (univoco) nel formato LLVVOO, ricavato dall'espressione LL*1000 + VV*100 + OO
- **COMANDO:** consentirà l'esecuzione dell'azione

CODICE AZIONE	CODIFICA	COMANDO
118	342957	usa panca
119	342947	Usa tapis/roulant
120	342999	Usa terminale

120	341099	Guarda terminale
121	342984	Usa scheda/schede

4.11 SALA GIOCHI

Il luogo “Sala Giochi” è stato inserito partendo dalla “La mia cabina” e passando ad est nel “Corridoio”, indirizzandosi a nord e scendendo raggiungiamo “Gli Uffici”, poi est raggiungendo “Negozio”, ad est raggiungendo “La banca” e nuovamente ad est raggiungendo il luogo desiderato, cioè “Sala Giochi”.

La sequenza di comandi per accedere al luogo Palestra è: est, Nord, scendi , est, est ed est.

Per effettuare l’integrazione della “Sala giochi”, a tale luogo sono state assegnate le seguenti caratteristiche univoche.

- **CODICE DEL LUOGO:** codice numerico che identifica il luogo
- **NOME DEL LUOGO:** stringa contenente il nome del luogo
- **CODICE DELLE DIREZIONI:** codice che consente di sapere le direzioni in cui è possibile andare (partendo dall’interno del luogo). Il codice è nel formato: NNSSEEOOUUDD, ovvero “Nord”, “Sud”, “Est”, “Ovest”, “Sali”, “Scendi”.

CODICE DEL LUOGO	NOME DEL LUOGO	CODICE DIREZIONI
36	Sala Giochi	000000150000

Come già accennato nel capitolo relativo all’analisi dell’integrazione, in questo luogo saranno presenti gli oggetti: una slot-machine, 5 euro, 10 euro, 20 euro, 50 euro, 100 euro.

4.11.1 Oggetto

- **Panca:** l’oggetto chiamato “Slot-Machine” è un oggetto non trasportabile. Il giocatore potrà usare la slot-machine e decidere quante volte giocare finché non decide di smettere o il denaro è insufficiente.
- **Euro:** l’oggetto chiamato “5 Euro” è un oggetto non trasportabile. È un possibile premo che si può ottenere giocando alla slot-machine.
- **10 Euro:** l’oggetto chiamato “10 Euro” è un oggetto non trasportabile. È un possibile premo che si può ottenere giocando alla slot-machine.
- **20 Euro:** l’oggetto chiamato “20 Euro” è un oggetto non trasportabile. È un possibile premo che si può ottenere giocando alla slot-machine.
- **50 Euro:** l’oggetto chiamato “50 Euro” è un oggetto non trasportabile. È un possibile premo che si può ottenere giocando alla slot-machine.
- **100 Euro:** l’oggetto chiamato “100 Euro” è un oggetto non trasportabile. È un possibile premo che si può ottenere giocando alla slot-machine.

All’oggetto sono state associate le seguenti caratteristiche:

- **CODICE DELL’OGGETTO:** codice (univoco) numerico che identifica l’oggetto.
- **NOME DELL’OGGETTO:** stringa contenente il nome dell’oggetto
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l’oggetto all’interno del gioco
- **CODICE MAPPA:** codice univoco che identifica la stanza in cui è presente l’oggetto. Il simbolo “-“ davanti al codice della mappa indica che quell’oggetto non è trasportabile

CODICE OGGETTO	NOME OGGETTO	VOCABOLO	CODICE MAPPA
86	Una slot-machine	Slot-machine	-36
90	5 Euro	5 Euro	-99
91	10 Euro	10 Euro	-99
92	20 Euro	20 Euro	-99

93	50 Euro	50 Euro	-99
94	100 Euro	100 Euro	-99

4.11.2 Azioni

Ad ogni azione sono state assegnate le seguenti caratteristiche univoche:

- **CODICE DELL'OGGETTO:** codice (univoco) che identifica il verbo.
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l'oggetto all'interno del gioco.
- **AZIONE:** l'azione che si intende eseguire.

CODICE OGGETTO	VOCABOLO	AZIONE
29	Usa	Giocare alla slot-machine

4.11.3 Comandi

Per ogni comando sono state assegnate le seguenti caratteristiche univoche:

- **CODICE AZIONE:** codice (univoco) che identifica il comando
- **CODIFICA:** codice (univoco) nel formato LLVVOO, ricavato dall'espressione LL*1000 + VV*100 + OO
- **COMANDO:** consentirà l'esecuzione dell'azione

CODICE AZIONE	CODIFICA	COMANDO
125	3600290086	usa slot-machine

4.12 BIBLIOTECA

Si modifica il numero di luoghi della prima riga del file impostandolo a 38

Aggiunta luogo Biblioteca con codice 37 e codice 28 per il comando scendi in scuola.

Aggiunta luogo Vetrina con codice 38 e codice 28 per il comando scendi in scuola.

Modifica del luogo Scuola inserendo il codice 37 (Biblioteca) per il comando sali.

CODICE DEL LUOGO	NOME DEL LUOGO	CODICE DIREZIONI
37	Biblioteca	0000000000028
38	Vetrina	0000000000028

4.12.1 Oggetti

Ad ogni oggetto sono state associate le seguenti caratteristiche:

- **CODICE DELL'OGGETTO:** codice (univoco) numerico che identifica l'oggetto.
- **NOME DELL'OGGETTO:** stringa contenente il nome dell'oggetto
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l'oggetto all'interno del gioco
- **CODICE MAPPA:** codice univoco che identifica la stanza in cui è presente l'oggetto. Il simbolo “-“ davanti al codice della mappa indica che quell'oggetto non è trasportabile

CODICE OGGETTO	NOME OGGETTO	VOCABOLO	CODICE MAPPA
70	Etichetta	Etichetta	-37
19	Vetrina	Vetrina	-37
90	Astronave	Astronave	38
91	Equipaggio	Equipaggio	38

4.12.2 Azioni

Ad ogni azione sono state assegnate le seguenti caratteristiche univoche:

- **CODICE DELL'OGGETTO:** codice (univoco) che identifica il verbo.
- **VOCABOLO:** parola chiave da utilizzare per poter interagire con l'oggetto all'interno del gioco.
- **AZIONE:** l'azione che si intende eseguire.

CODICE OGGETTO	VOCABOLO	AZIONE
8	Prendi	Prendi libro
9	Lascia	Restuisce libro
25	Leggi	Leggi libro

4.12.3 Comandi

- **leggi etichetta** che permette di visionare le informazioni sulla biblioteca.
- **apri vetrina** che permette di aprire la vetrina in cui sono contenuti i libri.
- **chiudi vetrina** che permette di chiudere la vetrina.
- **leggi astronave** che permette di leggere il libro astronave, se è presente nell'inventario.
- **leggi equipaggio** che permette di leggere il libro equipaggio, se è presente nell'inventario.

Per ogni comando sono state assegnate le seguenti caratteristiche univoche:

- **CODICE AZIONE:** codice (univoco) che identifica il comando
- **CODIFICA:** codice (univoco) nel formato LLVVOO, ricavato dall'espressione LL*100+ VV*100 + OO
- **COMANDO:** consentirà l'esecuzione dell'azione

CODICE AZIONE	CODIFICA	COMANDO
126	3700250070	leggi etichetta
127	3700220019	apri vetrina
128	3800930019	chiudi vetrina
129	0250090	leggi astronave
129	0250091	leggi equipaggio

4.13 BARI – ROMA – PISA

Partendo dalla “tua cabina”, per raggiungere gli oggetti è necessario spostarsi nella “cabina del secondo pilota” (Est-Nord-Ovest), prendere la chiave dell’auto o il ticket del bus o entrambi e successivamente raggiungere la “stazione di servizio” (Est-Sud-Sud-Sali), dove è possibile trovare i veicoli.

Arrivati alla “stazione di servizio”:

- digitando **Est** si raggiunge **Bari**
- da **Bari** digitando **Ovest** si ritorna alla stazione di servizio altrimenti digitando **Nord** si raggiunge **Roma**
- da **Roma** digitando **Sud** si ritorna a **Bari** altrimenti digitando **Nord** si raggiunge **Pisa**
- da **Pisa** digitando **Sud** si ritorna a **Roma**.

I luoghi sono definiti come segue:

- **Codice luogo:** codice univoco numerico che identifica il luogo
- **Nome luogo:** stringa contenente il nome del luogo

- **Codice delle direzioni:** codice che consente di sapere le direzioni in cui è possibile andare (partendo dal relativo luogo). Esso è composto da 6 sottostringhe che specificano rispettivamente le seguenti direzioni:
NNSSEEOOUUDD, ovvero “Nord”, “Sud”, “Est”, “Ovest”, “Sali (UP)”, “Scendi (DOWN)”.

Codice luogo	Nome luogo	Codice direzioni
45	Roma	474600000000
46	Bari	450000130000
47	Pisa	004500000000

Il raggiungimento dei suddetti luoghi mediante l'utilizzo di un veicolo comporta una perdita di tempo pari a 3, raggiungendoli “a piedi” si perderà 5.

Nei luoghi creati non ci sono né oggetti da prendere, né azioni particolari da compiere ricordando quindi che il loro scopo è far distrarre il giocatore.

4.13.1 Oggetti

Gli oggetti sono definiti come segue:

- **Codice oggetto:** codice univoco numerico che identifica l'oggetto
- **Nome oggetto:** stringa contenente il nome dell'oggetto
- **Codice luogo:** codice del luogo in cui si trova l'oggetto.

Codice oggetto	Nome oggetto	Codice luogo
114	ticket_bus	5 (Cabina secondo pilota)
115	autobus	13 (Stazione di Servizio)
116	chiave_auto	5 (Cabina secondo pilota)
117	automobile	13 (Stazione di Servizio)

4.13.2 Azioni

È stata implementata l'azione “guarda automobile” utilizzabile in qualsiasi luogo, in presenza dell'automobile e in possesso delle chiavi. Utilizzando questo comando, il giocatore verrà avvertito della presenza del bagagliaio (inizialmente vuoto) e gli verrà chiesto se vuole utilizzarlo per deporre o prelevare oggetti.

Le azioni sono definiti come segue:

- **Comando:** stringa composta da una o più parole necessarie per eseguire una determinata azione o scaturire un determinato evento.
- **Codifica:** codice numerico ottenuto dall'espressione matematica [Codice_Luogo * 10000 + Codice_Verbo * 100 + Codice_Oggetto] che permette di avere una codifica univoca
- **Codice azione:** codice identificativo dell'azione.

Comando	Codifica	Codice azione
guarda automobile	[00* 10000 + 10 * 100 + 117] = 1117	80

In questo comando possiamo notare come esso sia utilizzabile in qualsiasi luogo, dato dalla sostituzione di 00 a **Comando_Luogo**.

Il **Codice_Verbo** relativo al comando “*guarda*” è 10 (vocabolo già presente dal gioco), associato all’oggetto *automobile* che ha **Codice_Oggetto** 117.

4.14 MODIFICA CODICI LUOGHI – PALAGIANO MARCELLO

Sono state riscontrate delle incoerenze relative ai luoghi: Palestra, Sala Giochi, Biblioteca, Vetrina, Bari, Roma, Pisa, Luogotrasporto, Teletrasporto, Prima sala osservatorio, Seconda sala osservatorio, Terza sala osservatorio, Quarta sala osservatorio.

Per tale ragione si è deciso di modificare opportunamente i codici di tali luoghi come segue.

4.14.1 Modifica Mappa

NOME LUOGO	VECCHIO CODICE	NUOVO CODICE
Palestra	35	34
Sala giochi	36	35
Biblioteca	37	36
Vetrina	38	37
Roma	45	38
Bari	46	39
Pisa	47	40
Luogotrasporto	34	42
Teletrasporto	40	48

4.14.2 Modifica MappaOsservatorio

NOME LUOGO	VECCHIO CODICE	NUOVO CODICE
Luogotrasporto	35	43
Prima sala osservatorio	36	44
Seconda sala osservatorio	37	45
Terza sala osservatorio	38	46
Quarta sala osservatorio	39	47

4.14.3 Modifica MappaAliena

NOME LUOGO	VECCHIO CODICE	NUOVO CODICE
Sala teletrasporto astronave aliena	41	49
Corridoio sud astronave aliena	42	50
Corridoio nord astronave aliena	43	51
Sala comandi astronave aliena	44	52

4.15 MECCANICO

Il luogo Meccanico è raggiungibile, partendo dalla cabina di pilotaggio, andando a est (Corridoio), proseguendo verso sud (Estremità sud corridoio), salendo (Stazione di servizio) e, infine, procedendo verso sud.

La sequenza dei passi da seguire è la seguente:

- 1) Est
- 2) Sud
- 3) Sali
- 4) Sud

I luoghi all'interno del gioco sono costituiti da tre componenti:

- *Codice luogo*: codice numerico univoco, che lo distingue dagli altri luoghi presenti nella mappa;
- *Nome luogo*: stringa identificatrice del luogo;
- *Codice direzioni*: Stringa numerica, composta da sottostringhe che identificano i percorsi utilizzabili all'interno del luogo. Questa è formata da dodici cifre aventi il seguente significato:

NNSSEEEOOUUDD

dove N=Nord, S=Sud, E=Est, O=Ovest, U=Sali, D=Scendi.

Il luogo integrativo “Meccanico” è stato definito come segue:

CODICE DEL LUOGO	NOME DEL LUOGO	CODICE DIREZIONI
41	Meccanico	130000000000

4.15.1 Oggetti

I luoghi all'interno del gioco sono costituiti da quattro componenti:

- *Etichetta*: identifica il nome dell'oggetto;
- *Vocabolo*: parola chiave per interagire con l'oggetto all'interno del gioco;
- *Codice oggetto*: numero con cui è rappresentato l'oggetto all'interno del gioco;
- *Codice mappa*: codice univoco del luogo in cui è presente l'oggetto. Il segno “-“ indica che l'oggetto non può essere trasportato dall'avventuriero.

ETICHETTA	VOCABOLO	CODICE OGGETTO	CODICE MAPPA
un banco da lavoro	banco	43	-41
un pezzo di ricambio per astronave	ricambio	119	-41
una pila di batterie delle batterie scariche	pila	70	-41
una chiave_inglese	batterie	16	-41
un diario	chiave_inglese	118	41
un cartello	diario	57	-41
un computer	cartello	60	-41
un meccanico	computer	99	-41
	meccanico	73	-41

4.15.2 Azioni

Le azioni all'interno del gioco sono costituite da tre componenti:

- *Comando*: comando che il giocatore vuole fare eseguire al protagonista;

- **Codifica:** codice (univoco) nel formato
LLVVVVOOOO
dove:
LL è il codice del luogo nel quale il comando può essere impartito;
VV è il codice del verbo presente nel vocabolario;
OOOO è il codice dell'oggetto su cui è possibile eseguire l'azione.
- **Azione:** codice numerico dell'azione da eseguire.

Di seguito sono rappresentate le informazioni relative alle azioni integrate nel progetto base:

COMANDO	CODIFICA	AZIONE
guarda banco	4100100043	130
guarda ricambio	4100100119	131
guarda pila	4100100070	132
guarda batterie	4100100016	133
guarda chiave_inglese	100118	134
guarda diario	4100100057	135
leggi diario	4100250057	136
guarda cartello	4100100060	137
guarda computer	4100100099	138
parla meccanico	4100390073	139

4.16 CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”

4.16.1 Verbi “ruba” e “rubati”

La creazione dei verbi “ruba” e “rubati” avviene inserendo le due parole nel vocabolario del gioco.

Esse dovranno essere formate da:

- **Etichetta:** stringa che identifica il nome del verbo;
- **Codice:** codice numerico univoco che lo contraddistingue dagli altri verbi presenti nel gioco.

ETICHETTA	CODICE
ruba	50
rubati	200

A ciascun verbo da integrare sono state assegnate delle azioni per il loro richiamo con caratteristiche univoche che sono:

- **Codifica:** codice numerico ottenuto tramite l'espressione:

[Codifica comando = LL,VV,OO] in cui:

- LL = Codice del luogo in cui il comando può essere impartito.
- VV = Codice del verbo che indica una certa azione.

- OO = Codice dell'oggetto.
- Codice Azione:** codice univoco che rappresenta l'azione.

VERBO	CODIFICA	CODICE
ruba	100509999	-10
rubati	2000000	11

4.16.2 Personaggio “carabiniere”

Il personaggio “carabiniere” per le prime 10 azioni del giocatore resterà fisso nella stanza n° 2 (“Estremità nord del corridoio”), successivamente la stanza in cui verrà collocato sarà decisa da un numero calcolato randomicamente.

Nel momento in cui incontrerà il giocatore avrà il compito di perquisirlo: se trova uno dei tre oggetti che si possono rubare in suo possesso, lo porterà nel luogo “caserma”, altrimenti avviserà che non ha trovato nessun oggetto rubato in possesso del giocatore.

Per la sua creazione si sceglieranno:

- Nome del personaggio
- Frasi del personaggio
- Luogo in cui il personaggio si troverà inizialmente
- Codice univoco

4.16.3 Luogo “caserma”

Il luogo “caserma” è stato creato in modo tale da non essere raggiunto tramite altre stanze ma solo nel momento in cui il “carabiniere” arresta e ci porta il giocatore.

Al luogo da implementare sono stati assegnate delle caratteristiche univoche per facilitarne la lettura e la comprensione. Queste sono:

- **Etichetta:** stringa che identifica il nome del luogo;
- **Codice Mappa:** stringa che identifica i percorsi per arrivare e uscire dal luogo. Tale codice è formato da 12 caratteri aventi il seguente significato NNSSEEOUUDD
(N = Nord, S = Sud, E = est, O = ovest, U = Sali, D = Scendi);
- **Codice Luogo:** codice numerico univoco che lo contraddistingue dagli altri luoghi presenti nel gioco.

ETICHETTA	CODICE MAPPA	CODICE LUOGO
Caserma	030000000000	53

All'interno del luogo “caserma” non saranno disponibili oggetti poiché il giocatore non potrà interagirci ma dovrà solo rimanerci per qualche secondo.

4.17 ZAINO E VALIGIA

Per quanto riguarda gli oggetti Zaino e Valigia e per i frammenti verranno inseriti nuovi vocaboli, all'interno del vocabolario:

- Inserisci("aiuto", 259, vocabolario)
- Inserisci("help", 259, vocabolario)
- Inserisci("valigia", 155, vocabolario)
- Inserisci("zaino", 156, vocabolario)
- Inserisci("raccogli", 258, vocabolario)

E verranno inseriti i seguenti oggetti:

- Inserisci(Oggetto("una valigia", 155, 5, 0), oggetti)
- Inserisci(Oggetto("uno zaino", 156, 6, 0), oggetti)

Ogni oggetto sarà costituito dalla sua descrizione, codice, codice del luogo in cui è posto all'inizio del gioco e dal suo peso. Quest'ultimo attributo è stato aggiunto per gestire il peso massimo trasportabile nello zaino e nella valigia. Gli oggetti con peso uguale a zero sono gli oggetti il cui peso è trascurato nel gioco, mentre ad ogni oggetto trasportabile nello zaino o nella valigia è assegnato un peso maggiore di zero.

- L'oggetto Manuale avrà peso 1;
- L'oggetto Tuta avrà peso 2;
- L'oggetto Camice avrà peso 3;
- L'oggetto Casco avrà peso 3;

Tutti i pesi sono espressi in kg.

Verranno aggiunte le seguenti azioni:

- Inserisci(80155, -2, azioni) //azione prendi valigia
- Inserisci(80156, -2, azioni) //azione prendi zaino
- Inserisci(90155, 3, azioni) //azione lascia valigia
- Inserisci(90156, 3, azioni) //azione lascia zaino
- Inserisci(100155, 140, azioni) //azione guarda valigia
- Inserisci(100156, 141, azioni) //azione guarda zaino
- Inserisci(1550000, 143, azioni) //azione inventario valigia
- Inserisci(1560000, 144, azioni) //azione inventario zaino
- Inserisci(2590000, 145, azioni) //azione aiuto/help

E verranno modificate le seguenti azioni:

- Inserisci(2050, -71, azioni)
- Inserisci(2051, -71, azioni)
- Inserisci(2052, -71, azioni)

in:

- Inserisci (200050, -142)
- Inserisci (200051, -142)
- Inserisci (200052, -142)

Perciò ogni azione sarà costituita dal codice del comando e dal codice dell'azione da eseguire per quel comando.

Per rappresentare il codice del comando saranno necessarie 6 cifre del tipo LLVVOO. Le prime due indicheranno il codice del luogo (LL) in cui il comando è impartito, le due intermedie indicheranno il codice del verbo (VV) e le ultime due indicheranno il codice dell'oggetto (OO).

Azioni	LL	VV	OO	CODICE
Prendi valigia	00	08	15	-2
Prendi zaino	00	08	16	-2
Lascia valigia	00	09	15	3
Lascia zaino	00	09	16	3
Guarda valigia	00	10	15	69
Guarda zaino	00	10	16	70
Inventario valigia	00	15	00	72
Inventario zaino	00	16	00	73
Aiuto/help	00	29	00	74
Indossa/metti casco	00	20	50	-71
Indossa/metti tuta	00	20	51	-71
Indossa/metti camice	00	20	52	-71

Per quelle azioni in cui si prende un oggetto verrà usato il codice azione generale per "prendi oggetto", per quelle in cui si lascia un oggetto verrà usato il codice azione generale per "lascia oggetto".

I file del progetto base (Rosa Chiarappa) che hanno subito modifiche sono i seguenti:

- Astro.h
- Astro.cpp
- Gioco.h
- Gioco.cpp
- Mappa.h
- Interfaccia.cpp
- Oggetti.h
- Oggetti.cpp
- Oggetto.h
- Oggetto.cpp

Mentre sono stati aggiunti i seguenti file:

- `Zaino.h`
- `Valigia.h`

5 IMPLEMENTAZIONE

5.1 MUSEO

In seguito il codice relativo alle strutture e classi implementate:

5.1.1 Museo.h

```
#ifndef MUSEO_H
#define MUSEO_H

#include "Grafo.h"
#include "StanzaMuseo.h"

class Museo
{
public:
    Museo();
    void creamuseo();
    typename Grafo<StanzaMuseo,unsigned char>::nodo getstanzacorrente();
    void setstanzacorrente(typename Grafo<StanzaMuseo,unsigned char>::nodo);
    Lista<StanzaMuseo> getstanzeadiacenti();
    typename Grafo<StanzaMuseo,unsigned char>::nodo getiniziale();
    StanzaMuseo leggiStanza(typename Grafo<StanzaMuseo, unsigned char>::nodo);
    void scriviStanza(typename Grafo<StanzaMuseo, unsigned char>::nodo , const StanzaMuseo& );
private:
    typename Grafo<StanzaMuseo,unsigned char>::nodo stanzainiziale;
    typename Grafo<StanzaMuseo,unsigned char>::nodo stanzacorrente;
    Grafo<StanzaMuseo,unsigned char> grafomuseo;
```

```
    string titoli[25];
    string descrizioni[25];
};

//HEADER aggiunto da ANTONIO PASTORELLI
```

```
#endif // MUSEO_H
```

5.1.2 Museo.cpp

```
#include "Museo.h"
```

```
#include<iostream>
```

```
using namespace std;
```

```
Museo::Museo()
```

```
{
```

```
    creamuseo();
```

```
}
```

```
void Museo::creamuseo()
```

```
{
```

```
    titoli[0]="Galleria A: Percorso Preistorico.;"
```

descrizioni[0]="Il percorso Preistorico comprende una serie di esposizioni di notevole importanza. Per maggiori descrizioni sui reperti presenti, consultare la guida elettronica";

```
    titoli[1]="Neuropteris sp.-Felci fossili-Carbonifero-Francia";
```

descrizioni[1]="Queste piante raggiungevano un'altezza di cinque metri, con un tronco molto simile a quello delle attuali palme. Il fusto era formato in parte dalle basi foliari delle vecchie foglie, e possedeva radici aeree che crescevano in prossimità della base. Le fronde erano di diverso tipo: alcune erano dentellate, altre erano munite di foglioline arrotondate. A molte di queste foglie sono stati assegnati nomi diversi, come Neuropteris o Alethopteris. La prima era un tipo di foglia di grandi dimensioni, con un rachide percorso da striature longitudinali. In Alethopteris, invece, le venature erano poco visibili e quasi ad angolo retto, mentre il margine della foglia era dentellato.";

```
    titoli[2]="Allosaurus fragilis-Scheletro di Allosauro-Giurassico-Stati Uniti d'America";
```

descrizioni[2]="Allosaurus è un genere estinto di grande dinosauro teropode, vissuto tra i 155 e i 145 milioni di anni fa, durante il periodo Giurassico. I primi resti fossili furono ritrovati nel 1877, ad opera del paleontologo Othniel Charles Marsh, che ribattezzò i resti come Antrodemus. Essendo uno dei primi

dinosauri teropodi meglio conservati e piu' completi, questo animale ha piu' volte attirato l'attenzione di paleontologi e amanti dei dinosauri. L'Allosaurus era un predatore bipede di modeste dimensioni; il suo cranio era incredibilmente robusto e compatto e armato di una moltitudine di denti. La lunghezza di un esemplare adulto non doveva essere inferiore agli 8,5 metri di lunghezza, anche se alcuni resti frammentari suggeriscono dimensioni maggiori, con esemplari che avrebbero potuto raggiungere i 12 metri di lunghezza.";

titoli[3]="Copoliti di dinosauri - Epoche varie - Stati Uniti d'America";

descrizioni[3]="Il termine coprolite deriva dal greco kopros (sterco) e la-thos (pietra) e indica un escremento, prodotto da un animale vissuto nel passato, che si e' fossilizzato. Dall'analisi di questi reperti, si possono ricavare informazioni sulle abitudini alimentari e l'habitat nel quale l'animale viveva. Per esempio, la presenza di semi, foglie, corteccia o radici indica che l'escremento e' stato prodotto da un erbivoro, mentre se si individuano frammenti di ossa, artigli o tendini l'animale era un carnivoro.";

titoli[4]="Pterodactylus kochi-Rettile volante-Giurassico-Solnhofen (Germania)";

descrizioni[4]="Pterodactylus e' un estinto genere di pterosauro, i cui membri sono popolarmente chiamati "pterodattili". Attualmente, il genere contiene una singola specie, Pterodactylus antiquus, che oltre ad essere la specie tipo e' anche il primissimo genere di pterosauro mai rinvenuto. I principali ritrovamenti di resti fossili di questo animale sono stati rinvenuti principalmente, nei Calcari di Solnhofen, di Baviera, in Germania, risalenti alla fine del periodo Giurassico, circa 150,8-148-500 milioni di anni fa, anche se alcuni resti frammentari sono stati rinvenuti anche in altre aree in Europa e in Africa. Questo animale era un predatore che probabilmente si cibava soprattutto di pesci e piccoli invertebrati marini. Come tutti gli pterosauri, anche le ali dello Pterodactyluserano formate da una membrana di pelle che si estendeva dalla fine del quarto dito della "mano" fino agli arti posteriori. L'ala era supportata, ulteriormente, internamente da fibre di collagene ed esternamente da strutture cheratinose.";

titoli[5]="Tirannosauro vissuto nel Cretaceo superiore(tirannosauridi).";

descrizioni[5]="Il tirannosauro era un dinosauro vissuto nel Cretaceo superiore appartenente alla famiglia dei tirannosauridi. Visse in nordamerica, che anticamente era un continente isolato nominato Laramidia. Il Tyrannosaurus era molto piu' diffuso geograficamente degli altri tirannosauridi. I suoi fossili si trovano in una varietÃ di formazioni risalenti all' epoca Maastrichtiana del Cretaceo superiore, circa 68-66 milioni di anni fa. Fu una delle specie degli ultimi dinosauri non-aviani viventi quando si ebbe l' estinzione di massa del Cretaceo-Paleocene, che determino' la scomparsa dei dinosauri propriamente detti. ";

titoli[6]="Galleria B: Percorso Romano.";

descrizioni[6]="Come testimonianza di questo periodo storico potrete ammirare antichi affreschi. Per ulteriori dettagli, consultare la guida elettronica";

titoli[7]="Numa Pompilio istituisce il culto delle Vestali e dei sacerdoti(1636-1638)";

descrizioni[7]="Al centro della scena, sullo sfondo di un grandioso scorci architettonico, arde sull' altare il fuoco sacro che le Vestali dovevano custodire sempre acceso.";

titoli[8]="Ritrovamento della Lupa con Romolo e Remo (1596)";

descrizioni[8]="Faustolo scopre sotto i rami di un fico, sulla riva del Tevere, la Lupa che allatta Romolo e Remo. Nella figura della lupa e' evidente il richiamo alla Lupa capitolina conservata nel palazzo e simbolo della citta'.";

titoli[9]="Combattimento degli Orazi e Curiazi(1612-1613)";

descrizioni[9]="Episodio della guerra di Roma contro la vicina citta' di Albalonga che si concluse con un duello tra i rappresentanti di Roma, gli Orazi, e quelli di Albalonga, i Curiazi. Gli eserciti contendenti assistono alla scena finale del duello, quando l' ultimo degli Orazi sta per colpire l' ultimo degli avversari ";

titoli[10]="Ratto delle Sabine (1635-1636)";

descrizioni[10]="In primo piano e' il gruppo delle donne Sabine rapite dai Romani per popolare la citta' da poco fondata. L'affresco, eseguito dopo circa venti anni di interruzione, condivide con le ultime due scene una tecnica pittorica piu' rapida e sommaria, tipica della tarda maniera del Cavalier d' Arpino.";

titoli[11]="Battaglia di Tullo Ostilio contro i Veienti e i Fidenati(1597-1601)";

descrizioni[11]="Con vivacita' e' rappresentato un episodio della guerra di espansione intrapresa dai Romani contro le cittÃ vicine al tempo di Tullo Ostilio, terzo re di Roma. ";

titoli[12]="Galleria C: Ritrovamenti del periodo che va dal 200-300 a.C. ";

descrizioni[12]="Sono stati ritrovati numerosi reperti risalenti a questa epoca, alcuni dei quali sono presenti all'interno della nostra galleria. Per maggiori descrizioni, consultare la guida elettronica";

titoli[13]="Stele funeraria del tipo a "falsa porta" a nome di Sameri";

descrizioni[13]="La stele a " falsa porta " e' un elemento funerario tipico delle sepolture dell' Antico Regno ed e' costituita da due o piu' montanti laterali e da un architrave sotto cui e' scolpita una stuoia arrotolata a suggerire l' idea di una porta accessibile all' anima del defunto. Questa stele appartiene ad un funzionario di nome Sameri, membro di una famiglia di cortigiani molto vicina al faraone, e gli dedicata dal padre Urkaptah, che include nel dono anche la moglie e gli altri figli. Sameri e' rappresentato con la tecnica del rilievo ad incavo sull' architrave della "falsa porta", mentre siede in compagnia della madre Henutes davanti a una tavola ricolma di vasellame da mensa e di alimenti (pani, anatre giÃ spennate, tranci di carne bovina, etc.) utili per la sua sopravvivenza ultraterrena che l'iscrizione in caratteri geroglifici sottostante completa per forza magica. L'importanza del pane, quale alimento base della dieta egiziana, appare evidente, anche se non e' possibile differenziare le varie tipologie di pane elencate per qualitÃ di ingredienti, modalitÃ di impasto e di cottura.";

titoli[14]="Rilievo con scena di libagione ";

descrizioni[14]="Il rilievo, collocato in origine alle pareti di una tomba, mostra una scena di libagione. La **â€œsignora della casa**â€ Nubi, proprietaria della sepoltura, siede su una bassa sedia decorata con zampe di gatto mentre riceve l'offerta funeraria di acqua e di vino da sua figlia Kiki, in piedi davanti a lei. Altri cibi, tra i quali un mazzetto di porri e varie cucurbitacee, sono raccolti allâ€™interno di un profondo bacile alle spalle di Kiki allo scopo di nutrire in eterno la defunta. L'eleganza e la morbidezza delle figure, come la raffinatezza dei costumi rivelano l'agiatezza di vita della dama Nubi, tipica dell'Egitto del Nuovo Regno. Entrambe le donne indossano lunghi abiti la cui semplicitÃ contrasta con la resa accurata dei gioielli, visibili alle braccia, al collo e alle orecchie, e delle pesanti parrucche adorne di nastri coroncine di fiori e cono di unguento profumato.";

titoli[15]="Rilievo dalla tomba di Horemheb con scena di lavoro nei campi dell'oltretomba ";

descrizioni[15]="Il rilievo, proveniente da una delle tre cappelle di culto annesse alla tomba menfita del generale Horemheb, e' costituito da due frammenti parietali combacianti ed e' suddiviso in quattro fasce orizzontali scolpite a bassorilievo. E' probabile che quella in alto, di cui sopravvive solo la parte inferiore, contenesse l'omaggio di Horemheb alle divinitÃ funerarie. Nei due registri centrali, meglio conservati, Horemheb e' colto in momenti diversi: seduto in compagnia della sua **â€œanima**â€ akh dalle sembianze di uccello appollaiato su un trespolo davanti a una tavola piena di pani di varia forma, tranci di carne bovina e altri cibi destinati al pasto funebre del defunto; in piedi mentre governa alcuni buoi che calpestano un mucchio di spighe di cereale per separare i chicchi dalla pula; in piedi, dietro un aratro trainato da due buoi, intento a dissodare il terreno da coltivare. In basso, Horemheb e' nuovamente seduto davanti a una tavola stracolma di offerte e, a raccolto ultimato, riceve in dono alcuni mannelli di lino da parte di tre contadini. Tutte queste scene, ispirate al capitolo 110 del Libro dei Morti, ci mostrano Horemheb al lavoro nei campi dellâ€™oltretomba per garantirsi la sopravvivenza eterna. Il serpente ureo, simbolo di regalitÃ , che orna la sua fronte, fu scalpellato in un secondo momento, solo quando Horemheb divenne faraone dâ€™Egitto alla fine della XVIII dinastia.";

titoli[16]="Rilievo con la dea Renenutet";

descrizioni[16]="Il rilievo proviene dalla tomba di uno scriba di nome Amenemhat, vissuto agli inizi del Nuovo Regno, prima del faraone Akhenaton, durante il cui governo il nome del dio Amon fu spesso cancellato, come in questo caso. Sul rilievo sono raffigurate due tipiche attivitÃ agricole: nella parte alta, alcuni contadini puliscono il grano lanciandolo in aria con sessole di legno per separare i chicchi dalla pula, mentre un uomo offre loro da bere con profonde ciotole emisferiche, riempite in un grande vaso panciuto alle sue spalle. Nel registro inferiore si procede alla misurazione del cereale ormai pulito e accumulato di fronte alla dea delle messi Renenutet, in forma di serpente cobra. Scene di questo tipo, sia dipinte che a rilievo, sono piuttosto comuni nellâ€™arte egiziana a partire dall'Antico Regno.";

titoli[17]="Sarcofago a cassa a nome di Irinimenpu ";

descrizioni[17]="La decorazione principale di questo sarcofago destinato al corredo funerario di un egiziano di nome Irinimenpu consiste in una serie di pannelli a 'facciata di palazzo', un motivo ripreso dall'architettura funararia dell'Antico Regno, che rende il sarcofago simile a una dimora eterna. Su uno dei lati lunghi della cassa, in posizione centrale, sono raffigurate numerose offerte alimentari (pani, ortaggi, frutti, tranci di carni bovine, volatili giÃ spennati, contenitori per liquidi, etc.), perche' il defunto possa nutrirsi per forza magica nella vita ultraterrena. Sullo stesso lato, a una delle estremita', e' dipinta una porta chiusa, sopra la quale sono collocati due occhi che indicano la presenza all'interno della testa della

mummia e allo stesso tempo rappresentano una protezione magica per il corpo del defunto. L'ottimo stato di conservazione del legno e dei vivaci colori utilizzati per decorarlo si deve al clima caldo e asciutto dell'Egitto, oltre che alla collocazione originaria del sarcofago nell'ambiente protetto della sepolta.";

titoli[18]="Galleria D: Percorso sulla Magna Grecia";

descrizioni[18]="In questa galleria verrai guidato nell'esplorazione del percorso sulla Magna Grecia. La Magna Grecia è l'area geografica della penisola italiana meridionale che fu anticamente colonizzata dai Greci a partire dall'VIII secolo a.C.. Di questa civiltà possediamo numerose sculture e reperti. Per maggiori descrizioni sui reperti presenti, consultare la guida elettronica";

titoli[19]="Testa in marmo di Eracle";

descrizioni[19]="Probabilmente una copia del colosso bronzeo creato da Lisippo a Taranto alla fine del IV secolo AC. ";

titoli[20]="Stele figurata in marmo con defunto in nudità eroica. ";

descrizioni[20]="Raffigurato nell'atto di offrire una melagrana ad un serpente, simbolo funerario delle divinità infernali. ";

titoli[21]="Statua marmorea del II secolo d.C. di genio carpoforo (portatore di frutti). ";

descrizioni[21]="Proveniente dall'area delle Terme e relativo alla decorazione scultorea delle stesse. ";

titoli[22]="Specchio in argento, con doratura a caldo. ";

descrizioni[22]="È raffigurata l'immagine di una Musa, o di Afrodite, circondata da Erodi. Dalla tomba degli Ori di Canosa. ";

titoli[23]="I Bronzi di Riace";

descrizioni[23]="I Bronzi di Riace sono due statue di bronzo di provenienza greca o magnogreca o siceliota, databili al V secolo a.C. pervenute in eccezionale stato di conservazione. Le due statue, rinvenute il 16 agosto 1972 nei pressi di Riace, in provincia di Reggio Calabria sono considerate tra i capolavori scultorei più significativi dell'arte greca, e tra le testimonianze dirette dei grandi maestri scultori dell'età classica. Le ipotesi sulla provenienza e sugli autori delle statue sono diverse, ma non esistono ancora elementi che permettano di attribuire con certezza le opere ad uno specifico scultore. I Bronzi si trovano al Museo nazionale della Magna Grecia di Reggio Calabria, luogo in cui sono stati riportati il 12 dicembre 2015, dopo la rimozione e il soggiorno per tre anni (con annessi lavori di restauro) presso Palazzo Campanella, sede del Consiglio Regionale della Calabria a causa dei lavori di ristrutturazione dello stesso museo. I Bronzi sono diventati uno dei simboli della città stessa. ";

titoli[24]="Entrata museo delle scienze";

descrizioni[24]="Benvenuto nel museo delle scienze. Hai a disposizione una serie di gallerie visitabili, ognuna delle quali. Presenta diversi percorsi con varie tipologie di esposizioni. ";

StanzaMuseo stanza;

stanza.setdescrizione(descrizioni[24]);

stanza.settitolo(titoli[24]);

typename Grafo<StanzaMuseo,unsigned char>::nodo
nodo1,nodo2,nodo3,nodo4,nodo5,nodo6,nodo7,nodo8,nodo9,nodo10,nodo11,nodo12,nodo13,nodo14,

nodo15,nodo16,nodo17,nodo18,nodo19,nodo20,nodo21,nodo22,nodo23,nodo24,nodo0;

grafomuseo.insnodo(nodo24);

grafomuseo.scrivinodo(stanza,nodo24);

// setto la stanza iniziale e corrente nel grafo.

grafomuseo.insnodo(stanzacorrente);

grafomuseo.insnodo(stanzainiziale);

setstanzacorrente(nodo24);

grafomuseo.scrivinodo(stanza,stanzainiziale);

stanza.setdescrizione(descrizioni[0]);

stanza.settitolo(titoli[0]);

grafomuseo.insnodo(nodo0);

grafomuseo.scrivinodo(stanza,nodo0);

stanza.setdescrizione(descrizioni[6]);

stanza.settitolo(titoli[6]);

grafomuseo.insnodo(nodo6);

grafomuseo.scrivinodo(stanza,nodo6);

stanza.setdescrizione(descrizioni[12]);

```
stanza.settitolo(titoli[12]);
grafomuseo.insnodo(nodo12);
grafomuseo.scrivinodo(stanza,nodo12);
```

```
stanza.setdescrizione(descrizioni[18]);
stanza.settitolo(titoli[18]);
grafomuseo.insnodo(nodo18);
grafomuseo.scrivinodo(stanza,nodo18);
```

```
stanza.setdescrizione(descrizioni[1]);
stanza.settitolo(titoli[1]);
grafomuseo.insnodo(nodo1);
grafomuseo.scrivinodo(stanza,nodo1);
```

```
stanza.setdescrizione(descrizioni[2]);
stanza.settitolo(titoli[2]);
grafomuseo.insnodo(nodo2);
grafomuseo.scrivinodo(stanza,nodo2);
```

```
stanza.setdescrizione(descrizioni[3]);
stanza.settitolo(titoli[3]);
grafomuseo.insnodo(nodo3);
grafomuseo.scrivinodo(stanza,nodo3);
```

```
stanza.setdescrizione(descrizioni[4]);
stanza.settitolo(titoli[4]);
grafomuseo.insnodo(nodo4);
grafomuseo.scrivinodo(stanza,nodo4);
```

```
stanza.setdescrizione(descrizioni[5]);
stanza.settitolo(titoli[5]);
```

```
grafomuseo.insnodo(nodo5);
grafomuseo.scrivinodo(stanza,nodo5);

stanza.setdescrizione(descrizioni[7]);
stanza.settitolo(titoli[7]);
grafomuseo.insnodo(nodo7);
grafomuseo.scrivinodo(stanza,nodo7);

stanza.setdescrizione(descrizioni[8]);
stanza.settitolo(titoli[8]);
grafomuseo.insnodo(nodo8);
grafomuseo.scrivinodo(stanza,nodo8);

stanza.setdescrizione(descrizioni[9]);
stanza.settitolo(titoli[9]);
grafomuseo.insnodo(nodo9);
grafomuseo.scrivinodo(stanza,nodo9);

stanza.setdescrizione(descrizioni[10]);
stanza.settitolo(titoli[10]);
grafomuseo.insnodo(nodo10);
grafomuseo.scrivinodo(stanza,nodo10);

stanza.setdescrizione(descrizioni[11]);
stanza.settitolo(titoli[11]);
grafomuseo.insnodo(nodo11);
grafomuseo.scrivinodo(stanza,nodo11);

stanza.setdescrizione(descrizioni[13]);
stanza.settitolo(titoli[13]);
grafomuseo.insnodo(nodo13);
grafomuseo.scrivinodo(stanza,nodo13);
```

```
stanza.setdescrizione(descrizioni[14]);
stanza.settitolo(titoli[14]);
grafomuseo.insnodo(nodo14);
grafomuseo.scrivinodo(stanza,nodo14);

stanza.setdescrizione(descrizioni[15]);
stanza.settitolo(titoli[15]);
grafomuseo.insnodo(nodo15);
grafomuseo.scrivinodo(stanza,nodo15);

stanza.setdescrizione(descrizioni[16]);
stanza.settitolo(titoli[16]);
grafomuseo.insnodo(nodo16);
grafomuseo.scrivinodo(stanza,nodo16);

stanza.setdescrizione(descrizioni[17]);
stanza.settitolo(titoli[17]);
grafomuseo.insnodo(nodo17);
grafomuseo.scrivinodo(stanza,nodo17);

stanza.setdescrizione(descrizioni[19]);
stanza.settitolo(titoli[19]);
grafomuseo.insnodo(nodo19);
grafomuseo.scrivinodo(stanza,nodo19);

stanza.setdescrizione(descrizioni[20]);
stanza.settitolo(titoli[20]);
grafomuseo.insnodo(nodo20);
grafomuseo.scrivinodo(stanza,nodo20);

stanza.setdescrizione(descrizioni[21]);
```

```
stanza.settitolo(titoli[21]);
grafomuseo.insnodo(nodo21);
grafomuseo.scrivinodo(stanza,nodo21);

stanza.setdescrizione(descrizioni[22]);
stanza.settitolo(titoli[22]);
grafomuseo.insnodo(nodo22);
grafomuseo.scrivinodo(stanza,nodo22);

stanza.setdescrizione(descrizioni[23]);
stanza.settitolo(titoli[23]);
grafomuseo.insnodo(nodo23);
grafomuseo.scrivinodo(stanza,nodo23);

// Inserisco gli archi fra le stanze

grafomuseo.insarco(nodo0,nodo6);
grafomuseo.insarco(nodo6,nodo0);

grafomuseo.insarco(nodo6,nodo12);
grafomuseo.insarco(nodo12,nodo0);

grafomuseo.insarco(nodo12,nodo18);
grafomuseo.insarco(nodo18,nodo12);

grafomuseo.insarco(nodo0,nodo18);
grafomuseo.insarco(nodo18,nodo0);
```

```
grafomuseo.insarco(nodo24,nodo0);  
grafomuseo.insarco(nodo0,nodo24);
```

```
grafomuseo.insarco(nodo24,nodo6);  
grafomuseo.insarco(nodo6,nodo24);
```

```
grafomuseo.insarco(nodo24,nodo12);  
grafomuseo.insarco(nodo12,nodo24);
```

```
grafomuseo.insarco(nodo24,nodo18);  
grafomuseo.insarco(nodo18,nodo24);
```

```
grafomuseo.insarco(nodo0,nodo1);  
grafomuseo.insarco(nodo1,nodo0);
```

```
grafomuseo.insarco(nodo0,nodo2);  
grafomuseo.insarco(nodo2,nodo0);
```

```
grafomuseo.insarco(nodo3,nodo0);  
grafomuseo.insarco(nodo0,nodo3);
```

```
grafomuseo.insarco(nodo0,nodo4);  
grafomuseo.insarco(nodo4,nodo0);
```

```
grafomuseo.insarco(nodo0,nodo5);
grafomuseo.insarco(nodo5,nodo0);
```

```
grafomuseo.insarco(nodo6,nodo7);
grafomuseo.insarco(nodo7,nodo6);
```

```
grafomuseo.insarco(nodo6,nodo8);
grafomuseo.insarco(nodo8,nodo6);
```

```
grafomuseo.insarco(nodo6,nodo9);
grafomuseo.insarco(nodo9,nodo6);
```

```
grafomuseo.insarco(nodo6,nodo10);
grafomuseo.insarco(nodo10,nodo6);
```

```
grafomuseo.insarco(nodo6,nodo11);
grafomuseo.insarco(nodo11,nodo6);
```

```
grafomuseo.insarco(nodo12,nodo13);
grafomuseo.insarco(nodo13,nodo12);
```

```
grafomuseo.insarco(nodo12,nodo14);
grafomuseo.insarco(nodo14,nodo12);
```

```
grafomuseo.insarco(nodo12,nodo15);
grafomuseo.insarco(nodo15,nodo12);
```

```
grafomuseo.insarco(nodo12,nodo16);
grafomuseo.insarco(nodo16,nodo12);
```

```
grafomuseo.insarco(nodo12,nodo17);
grafomuseo.insarco(nodo17,nodo12);
```

```
grafomuseo.insarco(nodo18,nodo19);
grafomuseo.insarco(nodo19,nodo18);
```

```
grafomuseo.insarco(nodo18,nodo20);
grafomuseo.insarco(nodo20,nodo18);
```

```
grafomuseo.insarco(nodo18,nodo21);
grafomuseo.insarco(nodo21,nodo18);
```

```
grafomuseo.insarco(nodo18,nodo22);
grafomuseo.insarco(nodo22,nodo18);
```

```
grafomuseo.insarco(nodo18,nodo23);
grafomuseo.insarco(nodo23,nodo18);
```

```

}

typename Grafo<StanzaMuseo,unsigned char>::nodo Museo::getstanzacorrente()
{
    return(stanzacorrente);
}

typename Grafo<StanzaMuseo,unsigned char>::nodo Museo::getiniziale()
{
    return(stanzainiziale);
}

void Museo::setstanzacorrente(typename Grafo<StanzaMuseo,unsigned char>::nodo s)
{
    stanzacorrente = s;
}

Lista<typename Grafo<StanzaMuseo,unsigned char>::nodo> Museo::getstanzeadiacenti()
{
    return grafomuseo.adiacenti(stanzacorrente);
}

//Implementazione di Museom.h aggiunta da ANTONIO PASTORELLI

```

```

// Operatori Aggiunti da Graziano Montanaro

StanzaMuseo Museo::leggiStanza(typename Grafo<StanzaMuseo, unsigned char>::nodo stanza)
{
    return grafomuseo.legginodo(stanza);
}

void Museo::scriviStanza(typename Grafo<StanzaMuseo, unsigned char>::nodo stanza, const
StanzaMuseo& info)
{
    grafomuseo.scrivinodo(info, stanza);
}

```

5.1.3 Lista.h

```

#ifndef LISTA_H_
#define LISTA_H_

#include "Cella_L_DP.h"

template <class tipoelem>
class Lista{
public:
    typedef Cella_L_DP<tipoelem>* posizione;

    Lista();
    Lista(const Lista&);

    ~Lista();

    void crealista();

```

```

bool listavuota() const;
posizione primolista() const;
posizione succlista(posizione) const;
posizione preclista(posizione) const;
bool finelista(posizione) const;

tipoelem leggilista(posizione) const;

void inslista(tipoelem, posizione&);
void scrivilista(tipoelem, posizione);
void canclista(posizione&);

private:
    posizione lista;
};

template <class tipoelem>
Lista<tipoelem>::Lista()
{
    crealista();
}

template <class tipoelem>
Lista<tipoelem>::Lista(const Lista& b)
{
    crealista();
    typename Lista<tipoelem>::posizione ind = primolista(),ind2 = b.primolista();
    while (!b.finelista(ind2))
    {
        inslista(b.leggilista(ind2),ind);
        ind = succlista(ind);
        ind2 = b.succlista(ind2);
    }
}

```

```

template <class tipoelem>
Lista<tipoelem>::~Lista()
{
    posizione p = primolista();
    posizione temp;
    while(p->getSucc() != nullptr)
    {
        temp = p;
        p=p->getSucc();
        delete temp;
    }
}

template <class tipoelem>
void Lista<tipoelem>::crealista()
{
    lista = new Cella_L_DP<tipoelem>;
    lista->setSucc(nullptr);
    lista->setPrec(nullptr);
}

template <class tipoelem>
bool Lista<tipoelem>::listavuota() const
{
    return (lista->getSucc() == nullptr && lista->getPrec() == nullptr);
}

template <class tipoelem>
typename Lista<tipoelem>::posizione Lista<tipoelem>::primolista() const
{
    return lista;
}

template <class tipoelem>
typename Lista<tipoelem>::posizione Lista<tipoelem>::succlista(posizione p) const

```

```

{
    if(lista->getSucc() == nullptr)
        return p;
    else
        return p->getSucc();
}

template <class tipoelem>
typename Lista<tipoelem>::posizione Lista<tipoelem>::prelista(posizione p) const
{
    return p->getPrec();
}

template <class tipoelem>
bool Lista<tipoelem>::finelista(posizione p) const
{
    return (p->getSucc() == nullptr);
}

template <class tipoelem>
tipoelem Lista<tipoelem>::legglista(posizione p) const
{
    return p->getElem();
}

template <class tipoelem>
void Lista<tipoelem>::scrivilista(tipoelem e, posizione p)
{
    p->setElem(e);
}

template <class tipoelem>
void Lista<tipoelem>::inslista(tipoelem e, posizione& p)
{
}

```

```
typename Lista<tipoelem>::posizione temp;
```

```
temp = new Cella_L_DP<tipoelem>;
```

```
temp->setElem(e);
```

```
temp->setSucc(p);
```

```
temp->setPrec(p->getPrec());
```

```
if(p == primolista())
```

```
    lista = temp;
```

```
else
```

```
    p->getPrec()->setSucc(temp);
```

```
    p->setPrec(temp);
```

```
    p = temp;
```

```
}
```

```
template <class tipoelem>
```

```
void Lista<tipoelem>::canclista(posizione& p)
```

```
{
```

```
    posizione temp;
```

```
    temp = p;
```

```
    if(p == primolista())
```

```
{
```

```
        if(p->getSucc() != nullptr)
```

```
{
```

```
            lista = p->getSucc();
```

```
            lista->setPrec(nullptr);
```

```
}
```

```
}
```

```
else
```

```
{
```

```
    preclista(p)->setSucc(p->getSucc());
```

```
    suclist(p)->setPrec(preclista(p));
```

```

    }

p = p->getSucc();
delete temp;
}

```

#endif //LISTA_H_

5.1.4 Cella_L_DP.h

```

#ifndef CELLA_L_DP_H_
#define CELLA_L_DP_H_

template <class tipoelem>
class Cella_L_DP{
public:
    Cella_L_DP();
    ~Cella_L_DP();

    tipoelem getElem() const;
    void setElem(tipoelem);
    Cella_L_DP<tipoelem>* getSucc() const;
    void setSucc(Cella_L_DP<tipoelem>*);
    Cella_L_DP<tipoelem>* getPrec() const;
    void setPrec(Cella_L_DP<tipoelem>*);

private:
    tipoelem elem;
    Cella_L_DP<tipoelem>* succ;
    Cella_L_DP<tipoelem>* prec;
};


```

```

template <class tipoelem>
Cella_L_DP<tipoelem>::Cella_L_DP()
{
    succ = nullptr;
    prec = nullptr;
}

```

```
}
```

```
template <class tipoelem>
Cella_L_DP<tipoelem>::~Cella_L_DP(){}  
  
template <class tipoelem>
```

```
tipoelem Cella_L_DP<tipoelem>::getElem() const
{
    return elem;
}
```

```
template <class tipoelem>
void Cella_L_DP<tipoelem>::setElem(tipoelem e)
{
    elem = e;
}
```

```
template <class tipoelem>
Cella_L_DP<tipoelem>* Cella_L_DP<tipoelem>::getSucc() const
{
    return succ;
}
```

```
template <class tipoelem>
void Cella_L_DP<tipoelem>::setSucc(Cella_L_DP<tipoelem>* e)
{
    succ = e;
}
```

```
template <class tipoelem>
Cella_L_DP<tipoelem>* Cella_L_DP<tipoelem>::getPrec() const
{
    return prec;
}
```

```

template <class tipoelem>
void Cella_L_DP<tipoelem>::setPrec(Cella_L_DP<tipoelem>* e)
{
    prec = e;
}

#endif //CELLA_L_DP_H_

```

5.1.5 Grafo.h

```

#ifndef _GRAFO_H
#define _GRAFO_H

#include <iostream>
#include <cstdlib>
#include "Lista.h"
#include "Adiacenza.h"

//definizione della classe grafo (orientato, etichettato e pesato)
//realizzazione tramite vettore (di lunghezza maxnodi) con liste (monodirezionali dinamiche) di adiacenza

template<class tipoElem,class tipoPeso> class Grafo
{
public:

    //dichiarazioni di tipo
    typedef unsigned int nodo; //identificatore del nodo : il nodo è identificato da un intero (indice del vettore)
    typedef Adiacenza<nodo,tipoPeso> adiacente; //tipo dell'elemento che farà parte della lista di adiacenza
    //    adiacente = (rif.nodo adiac | peso arco)

    typedef struct //definizione dell'elemento del vettore
    {
        tipoElem etichetta; //etichetta del nodo di tipo tipoElem
        bool esiste; //campo booleano che indica se il nodo fa parte del grafo
        Lista<adiacente> adiac; //lista di adiacenza
    }
};


```

```

} cellaGrafo;

Grafo(); // costruttore
~Grafo(); // distruttore

unsigned int n_nodi();

//operatori di specifica
void creagrafo();
bool grafovuo() const;
void insnodo(nodo&);
void insarco(nodo,nodo);
void cancnodo(nodo);
void cancarco(nodo,nodo);
Lista<nodo> adiacenti(nodo) const;
bool esistenodo(nodo) const;
bool esistearco(nodo,nodo) const;
void scrivinodo(tipoElem,nodo);
tipoElem legginodo(nodo) const;
void scriviarco(tipoPeso,nodo,nodo);
tipoPeso leggiarco(nodo,nodo);

private:
cellaGrafo table[1000]; //Dimensiono il vettore
unsigned int maxnodi; //dimensione del vettore
unsigned int nelementi; //indica quanti nodi effettivamente ci sono nel grafo
nodo primolibero() const; //operatore ausiliare : restituisce la prima posizione libera del vettore (in modo da non
avere un vettore "sparso")
};


```

```

//n viene passato nel crea grafo

template<class tipoElem,class tipoPeso> Grafo<tipoElem, tipoPeso>::Grafo() //costruttore specifico
{
    creagrafo();
}

template<class tipoElem,class tipoPeso> Grafo<tipoElem, tipoPeso>::~Grafo() //distruttore
{

}

template<class tipoElem,class tipoPeso>
void Grafo<tipoElem, tipoPeso>::creagrafo() //crea il grafo
{
    maxnodi = 1000;
    nelementi=0; //dico che ci sono 0 nodi
    for (int i=0; i<1000; i++) //inizializzo il vettore mettendo a false l'appartenenza di tutti nodi
    {
        table[i].esiste=false;
    }
}

template<class tipoElem,class tipoPeso>
bool Grafo<tipoElem, tipoPeso>::grafovuento() const //restituisce true se il grafo è vuoto, false altrimenti
{
    return (nelementi==0);
}

template<class tipoElem,class tipoPeso>
void Grafo<tipoElem, tipoPeso>::insnodo(nodo &n) //inserisce il nodo che sarà identificato dall'indice n
{
    //passaggio del parametro per indirizzo perchè la variabile sarà modificata
    if (!esistenodo(n) && nelementi<1000) // precondizione nodo non appartenente
    {
        n=primolibero(); //la posizione in cui metterò il nodo del vettore sarà la prima libera
    }
}

```

```

table[n].esiste=true; //setto a true il suo campo esiste
nelementi++; //aumento il contatore di nodi del grafo
}

}

template<class tipoElem,class tipoPeso>
void Grafo<tipoElem, tipoPeso>::insarco(nodo n,nodo m) //inserisce l'arco che esce dal nodo n ed entra in m
{
if (esistenodo(n) && esistenodo(m) && !esistearco(n,m))//precondizione nodi appartenenti e arco non esistente
{
typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
adiacente temp; //creo l'elemento da inserire
temp.scrivinodo(m); //setto l'adiacente
table[n].adiac.inslista(temp,indice); //inserisco il nodo m negli adiacenti di n (in prima posizione)
}
}

template<class tipoElem,class tipoPeso>
void Grafo<tipoElem, tipoPeso>::cancnodo(nodo n) //elimina il nodo n
{
if (esistenodo(n))-----+
{
//|
if (table[n].adiac.listavuota())// 2      //|
{
//|
bool libero=true;           //|
int i=0;                   //|
while (i<maxnodi && libero)        //|
{
//|--- precondizione : nodo esistente 1),che non ha archi uscenti 2) nè entranti 3)
if (esistenodo(i))          //|
{
//|
libero=!esistearco(i,n);      //|
}
}
}
}

```

```

        }
                //|
        i++;           //|
    }
                //|
if (libero)// 3)-----+
{
    table[n].esiste=false; //adesso quel nodo non esiste più
    table[n].etichetta=NULL; //svuoto l'etichetta
    nlementi--; //e ho un nodo in meno nel grafo
}
}

}

template<class tipoElem,class tipoPeso>
void Grafo<tipoElem, tipoPeso>::cancarco(nodo n,nodo m) //elimina l'arco che esce da n ed entra in m
{
if (esistenodo(n) && esistenodo(m) && esistearco(n,m))//precondizione nodi appartenenti e arco esistente
{
    bool cancellato=false;
    typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
    while (!table[n].adiac.finlista(indice) && !cancellato) //scandisco la lista finchè non cancello l'elemento
    {
        if (table[n].adiac.leggilista(indice).legginodo()==m) //se trovo l'elemento lo cancello
        {
            table[n].adiac.canclista(indice); //elimino l'elemento
            cancellato=true;
        }
        else indice=table[n].adiac.succlista(indice);
    }
}
}

template<class tipoElem,class tipoPeso>
Lista<typename Grafo<tipoElem, tipoPeso>::nodo> Grafo<tipoElem, tipoPeso>::adiacenti(nodo n) const //restituisce la
lista di adiacenti di n

```

```

{
    Lista<nodo> lista; //lista da restituire
    adiacente temp; //comodo
    if (esistenodo(n)) // precondizione nodo appartenente al grafo
    {
        typename Lista<adiacente>::posizione indice=table[n].adiac.primolista(); //indice di scansione della lista di
        adiacenti (lista di adiacente)

        typename Lista<nodo>::posizione indice2=lista.primolista(); //indice di scansione della lista di adiacendi da
        restituire (lista di nodo)

        while (!table[n].adiac.finelista(indice)) //scansione della prima lista
        {
            temp=table[n].adiac.legglista(indice); //lettura di adiacente
            lista.inslista(temp.legginodo(),indice2); //inserisco (in coda) nella lista da restituire solo il riferimento al nodo
            adiacente (senza il peso dell'arco)

            indice=table[n].adiac.succlista(indice);
            indice2=lista.succlista(indice2);

        }
    }

    return(lista);
}

```

```

template<class tipoElem,class tipoPeso>
bool Grafo<tipoElem,tipoPeso>::esistenodo(nodo n) const //restituisce true se il nodo appartiene al grafo, false
altrimenti

```

```

{
    if (n<=maxnodi) return (table[n].esiste);
    else return false;
}

```

```

template<class tipoElem,class tipoPeso>
bool Grafo<tipoElem,tipoPeso>::esistearco(nodo n,nodo m) const //restituisce true se il nodo n ha un arco uscente verso
m, false altrimenti

```

```

{
    bool esiste=false;
    if (esistenodo(n) && esistenodo(m)) //precondizione nodi appartenenti al grafo
    {
        typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();

```

```

while (!table[n].adiac.finlista(indice) && !esiste) //scandisco la lista finchè non trovo l'elemento o è finita
{
    if (table[n].adiac.leggilista(indice).legginodo()==m) esiste=true;
    indice=table[n].adiac.succlista(indice);
}
return (esiste);
}

```

```

template<class tipoElem,class tipoPeso>
void Grafo<tipoElem, tipoPeso>::scrivinodo(tipoElem val,nodo n) //scrive l'etichetta del nodo n
{
    if (esistenodo(n)) //precondizione nodo appartenente al grafo
        table[n].etichetta=val;
}

```

```

template<class tipoElem,class tipoPeso>
tipoElem Grafo<tipoElem, tipoPeso>::legginodo(nodo n) const //restituisce l'etichetta del nodo n
{
    tipoElem e;
    if (esistenodo(n)) //precondizione nodo appartenente al grafo
        e=table[n].etichetta;
    return (e);
}

```

```

template<class tipoElem,class tipoPeso>
void Grafo<tipoElem, tipoPeso>::scriviarco(tipoPeso val,nodo n,nodo m) //scrive il peso dell'arco che va dal nodo n al
nodo m
{
    if (esistenodo(n) && esistenodo(m)) //precondizione nodi appartenenti al grafo (c'è anche arco appartenente ma
non conviene altrimenti c'è una scansione solo per vedere
    {
        //se esiste e poi si farebbe la seconda scansione per aggiornare
        bool aggiornato=false;                                //quindi si fa un'unica scansione in cui si ricerca e si
aggiorna)
        adiacente temp(m,val);
    }
}

```

```

typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();

while (!table[n].adiac.finlista(indice) && !aggiornato) //scandisco la lista finchè non trovo l'elemento o è finita
{
    if (table[n].adiac.leggilista(indice).legginodo()==m)
    {
        table[n].adiac.scrivilista(temp,indice);
        aggiornato=true;
    }
    indice=table[n].adiac.succlista(indice);
}
}

```

```

template<class tipoElem,class tipoPeso>

tipoPeso Grafo<tipoElem,tipoPeso>::leggiarco(nodo n,nodo m) //restituisce il peso dell'arco che va da n a m
{
    tipoPeso a;

    if (esistenodo(n) && esistenodo(m)) //precondizione nodi appartenenti al grafo (c'è anche arco appartenente ma
    non conviene altrimenti c'è una scansione solo per vedere
    {
        //se esiste e poi si farebbe la seconda scansione per la lettura                               //quindi si fa
        un'unica scansione in cui si ricerca e si legge)

        bool letto=false;

        typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();

        while (!table[n].adiac.finlista(indice) && !letto) //scandisco la lista finchè non trovo l'elemento o è finita
        {
            if (table[n].adiac.leggilista(indice).legginodo()==m)
            {
                a=table[n].adiac.leggilista(indice).leggipeso();
                letto=true;
            }
            indice=table[n].adiac.succlista(indice);
        }
    }
    return(a);
}

```

```

//operatori ausiliare

template<class tipoElem,class tipoPeso>
unsigned int Grafo<tipoElem, tipoPeso>::primolibero() const //restituisce la prima posizione del vettore libera
{
    nodo i=-1;
    bool libero=false;
    while (!libero) //scorre il vettore finchè non trova una posizione libera
    {
        //il controllo è solo su libero perchè so che mi fermerò per forza poichè questo metodo viene chiamato solo se il
        //vettore non è pieno
        i++;
        libero=!table[i].esiste;
    }
    return (i);
}

template<class tipoElem,class tipoPeso> unsigned int Grafo<tipoElem, tipoPeso>::n_nodi()
{
    return nelementi;
}

#endif

```

5.1.6 Adiacenza.h

```

#ifndef _ADIACENZA_H
#define _ADIACENZA_H

#include <iostream>
#include <cstdlib>

```

```

using namespace std;
84

```

```
//tipo dell'elemento che sarà usato nella lista di adiacenza del grafo pesato
```

```
template<class nodo,class tipoPeso> class Adiacenza
```

```
{
```

```
public:
```

```
//costruttori (generico di default)
```

```
Adiacenza();
```

```
Adiacenza(nodo,tipoS);
```

```
~Adiacenza();
```

```
//distruttore di default
```

```
//setter e getter
```

```
void scrivinodo(nodo);
```

```
nodo legginodo() const;
```

```
void scrivipeso(tipoPeso);
```

```
tipoPeso leggipeso() const;
```

```
private:
```

```
nodo adiacente; //riferimento al nodo adiacente
```

```
tipoPeso peso; //peso dell'arco
```

```
};
```

```
template <class nodo,class tipoPeso> Adiacenza<nodo,tipoS>::Adiacenza() //costruttore generico
```

```
{
```

```
}
```

```
template <class nodo,class tipoPeso> Adiacenza<nodo,tipoS>::Adiacenza(nodo n,tipoS p) //costruttore specifico
```

```
{
```

```
adiacente=n;
```

```

peso=p;
}

template <class nodo,class tipoPeso> Adiacenza<nodo,tipopeso>::~Adiacenza()
{
    //dtor
}

template <class nodo,class tipoPeso> void Adiacenza<nodo,tipopeso>::scrivinodo(nodo n)
{
    adiacente=n;
}

template <class nodo,class tipoPeso> void Adiacenza<nodo,tipopeso>::scrivipeso(tipopeso p)
{
    peso=p;
}

template <class nodo,class tipoPeso> nodo Adiacenza<nodo,tipopeso>::legginodo() const
{
    return(adiacente);
}

template <class nodo,class tipoPeso> tipopeso Adiacenza<nodo,tipopeso>::leggipeso() const
{
    return(peso);
}

//sovraffaccio output

```

```

template<class nodo,class tipoPeso> ostream& operator<<(ostream& os, const Adiacenza<nodo,tipopeso>& a)
{
    os<<"("<<a.legginodo()<<"|"<<a.leggipeso()<<")";
    return(os);
}

```

```
}
```

```
#endif
```

5.1.7 Modifiche relative ad Astro.cpp

```
void Astro::azione_96(){

    //system("cls"); windows
    system("clear"); //linux os //Modificato da ANTONIO PASTORELLI windows os

    int num_persone=(rand()+1)%11;
    bool uscita_museo=false;
    bool uscita_coda=false;
    string risposta_attesa; //input per la risposta
    int risposta_documento; //input per la risposta al documento

    while (!uscita_coda)
    {
        if (num_persone==0)
        {
            uscita_coda=true;
        }
        else
        {
            interfaccia.scrivi_parziale("Ci sono ");
            interfaccia.scrivi_parziale(num_persone);
            interfaccia.scrivi(" persone in coda per pagare il biglietto /n");
            interfaccia.scrivi("Il costo del biglietto e' 10 euro");
            interfaccia.scrivi("Desideri aspettare e pagare? (s/n):");
            cin >> risposta_attesa;

            if (risposta_attesa=="s" || risposta_attesa=="sÃ¬" || risposta_attesa=="si")
            {
                bool controlla=false;
                tempo=tempo-num_persone;
                uscita_coda=true;
                controlla=biglietto_museo();
            }
        }
    }
}
```

```

        if(controlla==false)
            uscita_museo=true;
    }
    else
        if (risposta_attesa=="n" || risposta_attesa=="no")
    {
        uscita_museo=true;
        uscita_coda=true;
    }
    else
    {
        interfaccia.scrivi("Input non compreso. Digitare correttamente le risposte!");
    }
}

```

//INIZIO MODIFICHE ANTONIO PASTORELLI

Museo museo;

Lista<typename Grafo<StanzaMuseo,unsigned char>::nodo> listastanze;

while (!uscita_museo)

{

copiaLista(listastanze,museo.getstanzeadiacenti());

interfaccia.scrivi(" -----");

cout<<endl<<museo.leggiStanza(museo.getstanzacorrente()).gettитolo()<<endl<<endl;

cout<<museo.leggiStanza(museo.getstanzacorrente()).getdescrizione()<<endl<<endl;

interfaccia.scrivi(" -----");

int risposta;

cout<<"Scegliere la stanza nella quale spostarsi:"<<endl;

typename Lista<Grafo<StanzaMuseo, unsigned char>::nodo>::posizione indice = listastanze.primolista();

int i = 0;

while(!listastanze.finlista(indice))

```

{
    cout<<i<<" - "<<museo.leggiStanza(listastanze.legglista(indice)).gettитolo()<<endl;
    i++;
    indice = listastanze.succlista(indice);
}

cout<<i<<" - Uscita dal Museo"<<endl;

i--;

cin>>risposta;

if (!cin)
{
    cin.clear();
    cin.ignore(256,'n');
    system("cls");
    //system("clear"); //linux os
    interfaccia.scrivi("Input errato");
    fflush(stdin);

}

else
{

    if((risposta>=0)&&(risposta<=i))
    {
        int j = 0;
        indice = listastanze.primolista();
        while(!(listastanze.finelista(indice)))
        {
            if(j == risposta)
            {
                museo.setstanzaCorrente(listastanze.legglista(indice));
                indice = listastanze.succlista(indice);
            }
        }
    }
}

```

```

        j++;
    }

    else
    {
        indice = listastanze.succlista(indice);
        j++;
    }

}

fflush(stdin);
system("cls");
//system("clear"); //linux os

}

else if(risposta == i+1)
{
    museo.setstanzacorrente(museo.getiniziale());
    uscita_museo = true;
    system("cls");
    //system("clear"); //linux os
    fflush(stdin);
}

else
{
    system("cls");
    //system("clear"); //linux os
    cout<<"Input errato"<<endl;
    fflush(stdin);
}

fflush(stdin);
}

```

5.2 SCUOLA E AULE

5.2.1 Astro.h

Aggiunta la riga di codice per le dichiarazioni dell'azione 97 cioè "lettura/guarda registro", per l'azione 98 cioè "guarda lavagna il gioco dell'impiccato", le azioni 100, 101 e 102 che sono rispettivamente "parla con il Preside", "parla con la maestra Clara" e "parla con la maestra Mara" e le azioni 103 e 104 che sono per "guarda cartina Galattica" e "guarda cartina Geografica".

```
//INIZIO MODIFICHE BASILE ANTONIO
void azione_97(); //lettura del registro
void azione_98(); //guarda lavagna gioco dell'impiccato
void azione_100(); //parla con il preside
void azione_101(); //parla con la maestra Clara
void azione_102(); //parla con la maestra Mara
void azione_103(); //guarda cartina Galattica
void azione_104(); //guarda cartina Geografica
//FINE MODIFICHE BASILE ANTONIO
```

5.2.2 Gioco.h

Sono stati aggiunti delle variabili booleane per far sì che i giochi si possano eseguire solo e soltanto una volta.

```
bool gioco_impiccato;
bool gioco_preside;
bool gioco_prof_mara;
bool gioco_prof_clara;
```

5.2.3 Astro.cpp

Attuate modifiche sulla procedura init_specifiche(), aggiungendo i vocaboli, gli oggetti e le azioni.

I vocaboli inseriti sono:

```
vocabolario.inserisci("scuola", 16);
```

```
vocabolario.inserisci("1A", 16);
vocabolario.inserisci("2A", 16);
vocabolario.inserisci("3A", 16);
vocabolario.inserisci("cartina", 60);
vocabolario.inserisci("banchi", 43);
vocabolario.inserisci("sedia", 56);
vocabolario.inserisci("registro", 88);
vocabolario.inserisci("lavagna", 90);
vocabolario.inserisci("cattedra", 70);
//inserimento di personaggi
vocabolario.inserisci("preside", 73);
vocabolario.inserisci("maestra", 73);
```

Gli oggetti inseriti sono:

```
//Oggetti inseriti nella scuola
oggetti.inserisci(Oggetto("una cartina Galattica", 60,-28));
oggetti.inserisci(Oggetto("il Preside", 73, -28));
//1A
oggetti.inserisci(Oggetto("la maestra Clara", 73, -29));
oggetti.inserisci(Oggetto("una cartina geografica", 60, -29));
oggetti.inserisci(Oggetto("dei banchi", 43, -29));
oggetti.inserisci(Oggetto("delle sedie", 56, -29));
oggetti.inserisci(Oggetto("una cattedra", 70, -29));
//2A
oggetti.inserisci(Oggetto("la maestra Mara", 73, -30));
oggetti.inserisci(Oggetto("dei banchi", 43, -30));
oggetti.inserisci(Oggetto("delle sedie", 56, -30));
oggetti.inserisci(Oggetto("un registro", 88, 30));
```

```
oggetti.inserisci(Oggetto("una cattedra", 70, -30));  
//3A  
oggetti.inserisci(Oggetto("dei banchi", 43, -31));  
oggetti.inserisci(Oggetto("delle sedie", 56, -31));  
oggetti.inserisci(Oggetto("una cattedra", 70, -31));  
oggetti.inserisci(Oggetto("una lavagna", 90, -31));
```

Le azioni inserite sono:

```
azioni.inserisci(2588,97); //leggi registro  
azioni.inserisci(1088, 97); //guarda registro  
azioni.inserisci(311090, 98); //guarda lavagna  
azioni.inserisci(283973, 100); //parla con il preside  
azioni.inserisci(293973, 101); //parla con la maestra Clara  
azioni.inserisci(303973, 102); //parla con la maestra Mara  
azioni.inserisci(281060, 103); //guarda cartina galattica  
azioni.inserisci(291060, 104); //guarda cartina geografica
```

Sono state inizializzate le variabili booleane a false:

```
gioco_impiccato = false;  
gioco_preside = false;  
gioco_prof_clara = false;  
gioco_prof_mara = false;
```

Nella procedura esegui_specifiche(int a, Mappa &M) sono stati aggiunti i seguenti case:

case 97:

```
azione_97();  
break;
```

case 98:

```

azione_98();
break;

case 100:
azione_100();
break;

case 101:
azione_101();
break;

case 102:
azione_102();
break;

case 103:
azione_103();
break;

case 104:
azione_104();
break;

```

Nello stesso file sono implementati le procedure void azione_97() (guarda lavagna), void azione_98() (gioco dell'impiccato), void azione_100() (parla con il preside), void azione_101() (parla con la maestra Clara), void azione_102() (parla con la maestra Mara), void azione_103() (guarda cartina Galattica), void azione_104() (guarda cartina geografica).

```

//guarda registro

void Astro::azione_97()
{
    interfaccia.scrivi("|-----|");

```

```

interfaccia.scrivi("----- Registro degli Alunni 2A -----");
interfaccia.scrivi("|- Data: 16/02/2017.....|");
interfaccia.scrivi("|- Albano Giuseppe.....ASSENTE|");
interfaccia.scrivi("|- Albino Gianni.....ASSENTE|");
interfaccia.scrivi("|- Franco Walter.....PRESENTE|");
interfaccia.scrivi("|- D'angelo Giuseppe.....ASSENTE|");
interfaccia.scrivi("|- Zagaria Marcello.....PRESENTE|");
interfaccia.scrivi("-----|");
}

//gioco dell'impiccato

void Astro::azione_98()
{
    system("cls");
    string risp;

    if(!gioco_impiccato)
    {
        do
        {
            interfaccia.scrivi("c'e' scritto qualcosa.....");
            interfaccia.scrivi("sembra che sia un gioco... ");
            interfaccia.scrivi("e' il gioco dell'impiccato!!!!");

            risp = interfaccia.leggi_stringa("Vuoi giocare al gioco dell'impiccato?
[Si/No]");

            }

        while(risp != "SI" && risp != "NO" && risp != "si" && risp != "no" &&
risp != "n" && risp != "s");
    }
}

```

```

if(risp == "si" || risp == "SI" || risp == "s")
{
    gioco_impiccato = true;
    string parola = "giocattolo";
    string parola_nascosta = "g****t***";
    string parola_indotrina;
    int num_tentativi = 3;
    int lunghezza = parola.length();
    bool indovina = false;
    bool controllo;
    bool trovato;
    char carattere;

do
{
    interfaccia.scrivi_parziale("Hai ");
    interfaccia.scrivi_parziale(num_tentativi);
    interfaccia.scrivi_parziale(" tentativi");
    interfaccia.a_capo();
    do
    {
        interfaccia.scrivi(parola_nascosta);
        risp = interfaccia.leggi_stringa("Vuoi indovinare la parola?");
    }
    while(risp != "SI" && risp != "NO" && risp != "si" && risp != "no"
&& risp != "n" && risp != "s");
    if(risp == "si" || risp == "s" || risp == "Si")

```

```

{
    indovina = true;
    parola_indotinata = interfaccia.leggi_stringa("Scrivi parola: ");
    if(parola == parola_indotinata)
    {
        interfaccia.scrivi("La parola e': giocattolo");
        interfaccia.scrivi("Hai Vinto!!!!");
        interfaccia.scrivi("Ti verra' incrementato il tempo di 40 punti!");
        tempo = tempo + 40;
    }
    else
    {
        interfaccia.scrivi("La parola da indovinare era: giocattolo");
        interfaccia.scrivi("Hai Perso!!!!");
        interfaccia.scrivi("Hai perso tempo ti verra' decrementato il tempo
di 40 punti!");
        tempo = tempo - 40;
    }
}
else
{
    trovato = false;
    carattere = interfaccia.leggi_carattere("Inserisci carattere: ");
    for(int i = 1; i <= lunghezza; i++)
    {
        if(parola_nascosta[i] == '*')
        {

```

```

        if(parola[i] == carattere)
        {
            parola_nascosta[i] = carattere;
            trovato = true;
        }
    }

if(!trovato)
{
    interfaccia.scrivi("Non hai indovinato nessun carattere");
    num_tentativi--;
}
else
{
    controllo = false;
    for(int i = 0; i <= lunghezza && !controllo; i++)
    {
        if(parola_nascosta[i] == '*')
            controllo = true;
    }
    if(!controllo)
    {
        interfaccia.scrivi("La parola e': giocattolo");
        interfaccia.scrivi("Hai Vinto!!!!");
        interfaccia.scrivi("Ti verra' incrementato il tempo di 40
punti!");
    }
}

```

```

tempo = tempo + 40;
indovina = true;
}

}

}

while(num_tentativi > 0 && !indovina);
if(num_tentativi == 0)
{
    interfaccia.scrivi("La parola da indovinare era: giocattolo");
    interfaccia.scrivi("Hai Perso!!!");
    interfaccia.scrivi("Hai perso tempo ti verra' decrementato il tempo di
40 punti!");
    tempo = tempo - 40;
}
else
{
    interfaccia.scrivi("Hai cose piu importanti da fare... salva l'astronave
Neutronia!!!");
}
else
{
    interfaccia.scrivi("Hai gia' giocato... salva l'astronave Neutronia!!!");
}

//parla con il preside
void Astro::azione_100()
{

```

```

string nome;
string accetta;
string risposta;

if(!gioco_preside)
{
    interfaccia.scrivi("Salve, benvenuto nella Scuola Elementare della
Neutronia");
    nome = interfaccia.leggi_stringa("Con chi ho il piacere di parlare?");
    interfaccia.scrivi_parziale("Quindi lei e' ");
    interfaccia.scrivi_parziale(nome);
    interfaccia.a_capo();

do
{
    accetta = interfaccia.leggi_stringa("Ho un problema da risolvere... Mi
puo' aiutare??? [Si/No]");
}

while(accetta != "si" && accetta != "s" && accetta != "Si" &&
      accetta != "no" && accetta != "n" && accetta != "No");

if(accetta == "si" || accetta == "s" || accetta == "Si")
{
    gioco_preside = true;
    interfaccia.scrivi("Benissimo!\n");
    interfaccia.scrivi("Sto cercando di risolvere un cruciverba... ");
    risposta = interfaccia.leggi_stringa("9 orizzontale.... un arma simile ad un
arco...");
```

```

if(risposta == "balestra")
{
    interfaccia.scrivi("Grazie mille!!!");
    interfaccia.scrivi("Il tuo tempo e' stato incrementato di 10 punti!!!");
    tempo = tempo + 10;
}

else
{
    interfaccia.scrivi("Hai Sbagliato!!!");
    interfaccia.scrivi("La parola era BALESTRA");
    interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti!!!");
    tempo = tempo - 10;
}

interfaccia.scrivi("Seconda domanda...");

risposta = interfaccia.leggi_stringa("1 orizzontale... L'ultima fa
traboccare il vaso: ");

if(risposta == "goccia")
{
    interfaccia.scrivi("Grazie mille!!!");
    interfaccia.scrivi("Il tuo tempo e' stato incrementato di 10 punti!!!");
    tempo = tempo + 10;
}

else
{
    interfaccia.scrivi("Hai Sbagliato!!!");
}

```

```

interfaccia.scrivi("La parola era Goccia");
interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti!!!!");
tempo = tempo - 10;
}

interfaccia.scrivi("Un'altra domanda... ");
risposta = interfaccia.leggi_stringa("10 verticale.... Interruttore per...
liquidi: ");
if(risposta == "rubinetto")
{
    interfaccia.scrivi("Grazie mille!!!!");
    interfaccia.scrivi("Il tuo tempo e' stato incrementato di 10 punti!!!!");
    tempo = tempo + 10;
}
else
{
    interfaccia.scrivi("Hai Sbagliato!!!!");
    interfaccia.scrivi("La parola era Rubinetto");
    interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti!!!!");
    tempo = tempo - 10;
}

}

else
{
    interfaccia.scrivi_parziale("Ciao ");
    interfaccia.scrivi_parziale(nome);
}

```

```

    interfaccia.a_capo();

}

}

else

    interfaccia.scrivi("Hai gia' parlato con il preside");

}

//parla con la maestra Clara

void Astro::azione_101()

{

    if(!gioco_prof_clara)

    {

        string risp;

        do

        {

            interfaccia.scrivi("Salve sono la maestra Clara");

            risp = interfaccia.leggi_stringa("Insegno storia, vuoi fare un test?

[Si/No]");



        }

        while(risp != "Si" && risp != "No" && risp != "s" && risp != "n" &&

             risp != "si" && risp != "no");



        if(risp == "Si" || risp == "s" || risp == "si")

        {

            string risposta;

            gioco_prof_clara = true;

```

```

interfaccia.scrivi("Rispondi alle seguenti domande:");
risposta = interfaccia.leggi_stringa("Prima domanda: Quando cadde
l'Impero Romano d'Occidente?");
if(risposta == "476 d.C.")
{
    tempo = tempo + 5;
    interfaccia.scrivi("Risposta Esatta");
    interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti");
}
else
{
    tempo = tempo - 10;
    interfaccia.scrivi("Risposta sbagliata");
    interfaccia.scrivi("il tuo tempo e' stato decrementato di 10 punti");
}
interfaccia.a_capo();
risposta = interfaccia.leggi_stringa("Seconda domanda: Quando venne
scoperta l'America?");
if(risposta == "1492")
{
    tempo = tempo + 5;
    interfaccia.scrivi("Risposta Esatta");
    interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti");
}
else
{
    tempo = tempo - 10;
}

```

```

    interfaccia.scrivi("Risposta sbagliata");
    interfaccia.scrivi("il tuo tempo e' stato decrementato di 10 punti");
}

interfaccia.a_capo();
risposta = interfaccia.leggi_stringa("Terza Domanda: Quando esplose la
prima Guerra Mondiale?");
if(risposta == "1914")
{
    tempo = tempo + 5;
    interfaccia.scrivi("Risposta Esatta");
    interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti");
}
else
{
    tempo = tempo - 10;
    interfaccia.scrivi("Risposta sbagliata");
    interfaccia.scrivi("il tuo tempo e' stato decrementato di 10 punti");
}
else
    interfaccia.scrivi("Ok ci vediamo!");
}

else
    interfaccia.scrivi("Hai gia' parlato con la maestra Clara");
}

//parla con la maestra mara

```

```

void Astro::azione_102()
{
    if(!gioco_prof_mara)
    {
        string risp;
        do
        {
            interfaccia.scrivi("Salve sono la maestra Mara");
            risp = interfaccia.leggi_stringa("Insegno matematica, Vuoi fare un test?
[Si/No]");
        }
        while(risp != "Si" && risp != "No" && risp != "s" && risp != "n" &&
              risp != "si" && risp != "no");

        if(risp == "Si" || risp == "s" || risp == "si")
        {
            gioco_prof_mara = true;
            int num_esatte;
            string risp1, risp2, risp3;
            interfaccia.scrivi("Risolvi le seguenti espressioni: ");
            interfaccia.scrivi("1. [(3*4)/2+5*6-(10/5)*6]");
            interfaccia.scrivi("2. [6*7-18*(5-12)]");
            interfaccia.scrivi("3. [(4^2/8)*7+(2^3)*4]");
            interfaccia.a_capo();

            risp1 = interfaccia.leggi_stringa("Risultato della prima espressione: ");

```

```

if(risp1 == "24")
{
    interfaccia.scrivi("Risposta Corretta! Bravo!!!");
    interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti.");
    tempo = tempo + 5;
}
else
{
    interfaccia.scrivi("Risposta Sbagliata!");
    interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti");
    tempo = tempo - 10;
}

```

risp2 = interfaccia.leggi_stringa("Risultato della seconda espressione: ");

```

if(risp2 == "168")
{
    interfaccia.scrivi("Risposta Corretta! Bravo!!!");
    interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti.");
    tempo = tempo + 5;
}
else
{
    interfaccia.scrivi("Risposta Sbagliata!");
    interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti");
    tempo = tempo - 10;
}

```

```
risp3 = interfaccia.leggi_stringa("Risultato della terza esprezzione:  ");

if(risp3 == "46")
{
    interfaccia.scrivi("Risposta Corretta! Bravo!!!");
    interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti.");
    tempo = tempo + 5;
}
else
{
    interfaccia.scrivi("Risposta Sbagliata!");
    interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti");
    tempo = tempo - 10;
}
else
{
    interfaccia.scrivi("Ok ci vediamo");
}
}

//guarda cartina galattica
void Astro::azione_103()
{
```

```

    interfaccia.scrivi("e' una cartina che rappresenta il Sistema Solare");
}

//guarda cartina geografica

void Astro::azione_104()
{
    interfaccia.scrivi("cartina che rappresenta la penisola Italiana");
}

```

5.2.4 Mappa.nav

Per poter aggiungere i luoghi “scuola”, “1A”, “2A”, “3A”, è stato necessario modificare il numero dei luoghi da 27 a 31, aggiornando le righe di navigazione per permettere al luogo di essere raggiunto. La “scuola” è stata inserita a nord della cabina di pilotaggio, la “classe 1A” è stata inserita a est della “scuola”, la “classe 2A” è stata inserita a nord della scuola e la “classe 3A” è stata inserita a ovest della scuola.

Per poter raggiungere la scuola è stata la riga di comando del luogo “cabina di pilotaggio”:

1,Nella cabina di pilotaggio,280000000002,via 1,1,1

È stata aggiunta la riga di comando per il luogo “Scuola”:

28,Nella Scuola,300129310000,via 1,1,1

Sono state aggiunte le righe di comando per raggiungere o abbandonare la “Scuola”:

1,28 via 1,nord,2,2,1,1
 28,1 via 1,sud,2,2,1,1

Righe di comando per aggiungere le classi:

29,Nella Classe 1A,000000280000,via 1,1,1
 30,Nella Classe 2A,002800000000,via 1,1,1
 31,Nella Classe 3A,000028000000,via 2,2,1

Righe di comando per raggiungere o abbandonare le classi:

28,29, via 1,est,1,1,1,
29,28, via 1,ovest,1,1,1
28,30, via 1,nord,1,1,1
30,28, via 1,sud,1,1,1
28,31, via 2,ovest,1,1,1
31,28, via 1,est,2,2,1

5.2.5 Aggiunta dei file nella cartella descrizioni

Nella cartella “Descrizioni” è stato aggiunto il file relativo alla “Scuola” (28.txt), il file relativo alla “classe 1A” (29.txt), il file relativo alla “classe 2A” (30.txt) e il file relativo alla “classe 3A” (31.txt).

La descrizione del luogo “Scuola” (luogo 28)

Nella Scuola

Di nuovo nella Scuola

Nuovamente nella Scuola

Ancora nella Scuola

La descrizione del luogo “Classe 1A” (luogo 29)

Nella Scuola

Di nuovo nella Scuola

Nuovamente nella Scuola

Ancora nella Scuola

La descrizione del luogo “Classe 2A” (luogo 30)

nella Classe 2A

di nuovo nella Classe 2A

nuovamente nella Classe 2A

ancora nella Classe 2°

La descrizione del luogo “Classe 3A” (luogo 31)

nella Classe 3A

sei di nuovo nella Classe 3A

nuovamente nella Classe 3A

ancora nella Classe 3A

5.3 ARCHIVIO E MUSEO (VECCHIA VERSIONE DEL MUSEO)

5.3.1 Astro.h

Aggiunta la riga di codice per le dichiarazioni dell'azione 91 ovvero l'uso del computer nell'archivio, per l'azione 92 ovvero la lettura del file 12, per l'azione 93 ovvero la lettura del file 13, per l'azione 94 ovvero la lettura del file 15, per l'azione 95 ovvero la lettura del file 21 e per l'azione 96 ovvero l'ingresso nel museo.

E' stata modificata la numerazione delle azioni per evitare conflitti con quelle già esistenti.

void azione_91();//uso del computer nell'archivio

void azione_92();//lettura file 12

void azione_93();//lettura file 13

void azione_94();//lettura file 15

void azione_95();//lettura file 21

void azione_96();//ingresso nel museo

5.3.2 Gioco.h

Aggiungi i prototipi delle funzioni implementate in Gioco.cpp

void Aggiorna_Tempo_bonus();//bonus tempo

bool biglietto_museo();//acquisto biglietto

void mostra_galleria_a();//esplorazione galleria a

void mostra_galleria_b();//esplorazione galleria b

void mostra_galleria_c();//esplorazione galleria c

void mostra_galleria_d();//esplorazione galleria d

5.3.3 Astro.cpp

Inserimento del vocabolo "file12", "file13", "file15", "file21", "computer", "museo", "entra nel" all'interno del vocabolario del gioco.

I codici scelti per i vocaboli sono rispettivamente 37,57,58,71,99,16,74.

Il vocabolo computer viene usato come sinonimo del vocabolo terminale.

vocabolario.inserisci("file12",37);

vocabolario.inserisci("file13",57);

vocabolario.inserisci("file15",58);

vocabolario.inserisci("file21",71);

vocabolario.inserisci("computer",99);

vocabolario.inserisci("museo",16);//definisco il vocabolo museo

vocabolario.inserisci("entra nel",74);//duplicato di "entra"

Sono stati inseriti gli oggetti relativi ai documenti ed il museo:

oggetti.inserisci(Oggetto("file12",37, -26));

oggetti.inserisci(Oggetto("file13",57, -26));

oggetti.inserisci(Oggetto("file15",58, -26));

oggetti.inserisci(Oggetto("file21",71, -26));

oggetti.inserisci(Oggetto("un computer",99, -26));

oggetti.inserisci(Oggetto("il museo",16, -16));

Nella sezione relativa alle azioni non è stato necessario aggiungere il comando che permette al giocatore di utilizzare il computer (“usa computer” oppure “usa terminale” oppure “usa calcolatore”) in quanto già presente un’implementazione: È stata aggiunta soltanto l’azione per utilizzare il calcolatore.

Successivamente è stato necessario aggiungere le azioni che avrebbero permesso al giocatore di leggere i file stampati e posseduti nell’inventario.

Nell’integrare il progetto di Crocco in quello di Galeandro è stato opportuno modificare la parte del codice relativo al luogo in cui viene svolta l’azione. Il luogo 24 usato per l’archivio era occupato dall’ufficio postale mentre il luogo 25 usato per il Museo era occupato dall’Aula. E’ stata quindi modificata la numerazione in modo da aggiungere al numero 26 l’archivio e al numero 27 il museo.

azioni.inserisci(262999, 91);//uso del computer

azioni.inserisci(262537, 92);//leggi file12

azioni.inserisci(262557, 93);//leggi file13

azioni.inserisci(262558, 94);//leggi file15

azioni.inserisci(262571, 95);//leggi file21

azioni.inserisci(277416, 96); //azione entrata museo

Sono stati di conseguenza aggiunti I seguenti case:

case 91:

azione_91();

break;

case 92:

azione_92();

break;

case 93:

```
azione_93();
```

```
    break;
```

case 94:

```
azione_94();
```

```
    break;
```

case 95:

```
azione_95();
```

```
    break;
```

case 96:

```
azione_96();
```

```
    break;
```

Sono state inserite all'interno del case le nuove azioni numerate da 91 a 95 relative all'utilizzo del computer e alla visualizzazione dei file.

Perciò l'azione 91 corrisponde ad "usa computer" (o eventuali sinonimi) nel luogo in cui verrà utilizzato (nell'archivio) identificato dal numero 26.

Le azioni che vanno da 92 a 95 sono relative alla lettura dei file. Saranno richiamate dal verbo "leggi" seguito dal nome dell'oggetto (file). Il file dovrà essere scaricato dal computer per poterlo leggere.

L'azione 96 corrisponde ad "entra nel museo" e consentirà l'accesso al museo previo acquisto del biglietto ed attesa della coda.

```
void Astro :: azione_91 () {
```

```
    system("cls");
```

```
    int num_persone=rand()%11;
```

```
    bool uscita_computer=false;
```

```
    bool uscita_coda=false;
```

```
    string risposta_attesa; //input per la risposta
```

```
    int risposta_documento; //input per la risposta al documento
```

```
    while (!uscita_coda)
```

```
    {
```

```
        if (num_persone==0)
```

```

{
    uscita_coda=true;
}

else

{
    interfaccia.scrivi_parziale("Ci sono ");
    interfaccia.scrivi_parziale(num_persone);
    interfaccia.scrivi(" in coda che vogliono utilizzare il computer");
    interfaccia.scrivi("Desideri aspettare (s/n):");
    cin >> risposta_attesa;

    if (risposta_attesa=="s" || risposta_attesa=="sÃ¬" || risposta_attesa=="si")
    {
        tempo=tempo-num_persone;
        uscita_coda=true;
    }

    else

    if (risposta_attesa=="n" || risposta_attesa=="no")
    {
        uscita_computer=true;
        uscita_coda=true;
    }

    else

    {
        interfaccia.scrivi("Input non compreso. Digitare correttamente le
        risposte!");
    }
}

```

```

while (!uscita_computer)

{
    // Menu di scelta 0,1, ...

    system("cls");

    interfaccia.scrivi(" -----");
    interfaccia.scrivi("-----");

    interfaccia.scrivi("|Sistema Archivio 2.0 |");
    interfaccia.scrivi("|Benvenuto Comandante nel sistema di archivio |");
    interfaccia.scrivi("dell'astronave. |");

    interfaccia.scrivi("|A causa dei gravi danni subiti, l'integrita' dell'archivio e' |");
    interfaccia.scrivi(" compromessa. |");

    interfaccia.scrivi("|Il sistema e' stato in grado di ripristinare solo alcuni file. |");
    interfaccia.scrivi(" |");

    interfaccia.scrivi("|Inoltre vi e' un problema nella visualizzazione a schermo dei |");
    interfaccia.scrivi(" file. |");

    interfaccia.scrivi("|I file possono pero' essere stampati senza problemi |");
    interfaccia.scrivi(" |");

    interfaccia.scrivi("|Per ritirare il documento digitare il numero corrispondente. |");
    interfaccia.scrivi(" |");

    interfaccia.scrivi("|Attenzione: problemi tecnici rallenteranno il processo di |");
    interfaccia.scrivi(" stampa. |");

    interfaccia.scrivi("|Per uscire digitare il numero corrispondente. |");
    interfaccia.scrivi(" |");

    interfaccia.scrivi("|1 <- File12 |");
    interfaccia.scrivi("|2 <- File13 |");
    interfaccia.scrivi("|3 <- File15 |");
    interfaccia.scrivi("|4 <- File21 |");
}

```

```

interfaccia.scrivi("|5 <- Uscire dal computer
|");
interfaccia.scrivi(" -----\n-----\n");
interfaccia.scrivi_parziale("Mosse rimaste: ");
interfaccia.scrivi_parziale(tempo);
interfaccia.scrivi_parziale("\n");
interfaccia.scrivi_parziale("Risposta: ");

cin >>risposta_documento;
switch (risposta_documento) {
    case 1:
        //file12
        if (oggetti.get_oggetto(37).get_luogo() != 0)
        {
            oggetti.set_luogo(37,0);
            interfaccia.scrivi("File12 ritirato!");
            tempo=tempo-(rand()%6+2);
            system("pause");
        }
        else
        {
            interfaccia.scrivi("File gia' ritirato e disponibile per la lettura!");
            tempo=tempo-1;
            system("pause");
        }
        break;
}

```

case 2:

```
//file13

if (oggetti.get_oggetto(57).get_luogo() != 0)

{
    oggetti.set_luogo(57,0);
    interfaccia.scrivi("File13 ritirato!");
    tempo=tempo-(rand()%6+2);
    system("pause");
}

else

{
    interfaccia.scrivi("File gia' ritirato e disponibile per la lettura!");
    tempo=tempo-1;
    system("pause");

}
```

break;

case 3:

```
//file15

if (oggetti.get_oggetto(58).get_luogo() != 0)

{
    oggetti.set_luogo(58,0);
    interfaccia.scrivi("File15 ritirato!");
    tempo=tempo-(rand()%6+2);
    system("pause");
}

else

{
```

```

interfaccia.scrivi("File gia' ritirato e disponibile per la lettura!");
tempo=tempo-1;
system("pause");
}

break;

case 4:
//file21

if (oggetti.get_oggetto(71).get_luogo() != 0)
{
    oggetti.set_luogo(71,0);
    interfaccia.scrivi("File21 ritirato!");
    tempo=tempo-(rand()%6+2);
    system("pause");
}

else
{
    interfaccia.scrivi("File gia' ritirato e disponibile per la lettura!");
    tempo=tempo-1;
    system("pause");
}

break;

case 5:
uscita_computer=true;
system("cls");
break;

default:
interfaccia.scrivi("Comando non riconosciuto... stai per uscire!");
uscita_computer=true;

```

```

    system("pause");
}

}

}

```

L'azione 92 è relativa alla lettura del "file12" il quale contiene in maniera indiretta un utile suggerimento sul corretto utilizzo del compartimento stagno.

void Astro :: azione_92 ()

```
{

```

```
system("cls");
```

```
if(oggetti.get Oggetto(37).get Luogo() != 0)
```

```
{

```

```
    interfaccia.scrivi("Puoi leggere i file solo dopo averli scaricati dal terminale  
dell'archivio");
```

```
}
```

else

```
{
```

```
    interfaccia.scrivi("File12");
```

```
    interfaccia.scrivi("Le simulazioni di volo hanno evidenziato diversi problemi.");
```

```
    interfaccia.scrivi("L'equipaggio non e' a conoscenza di particolari rischi derivanti  
dal scorretto utilizzo dei sistemi di bordo.");
```

```
    interfaccia.scrivi("Per aprire il compartimento stagno e' necessario premere il  
pulsante rosso.");
```

```
    interfaccia.scrivi("Per richiudere il compartimento stagno bisogna premere il  
verde.");
```

```
    interfaccia.scrivi("Pare che i cadetti abbiano fatto l'esatto opposto.");
```

```
}
```

```
}
```

L'azione 93 è relativa alla lettura del "file13" il quale, in realtà, non contiene alcuna informazione utile ai fini del gioco.

void Astro::azione_93(){

119

```

system("cls");
if(oggetti.get_oggetto(57).get_luogo() != 0)
{
    interfaccia.scrivi("Puoi leggere i file solo dopo averli scaricati dal terminale dell'archivio");
}
else
{
    interfaccia.scrivi("File13");
    interfaccia.scrivi("La forza di gravita' e' spesso sottovalutata.");
    interfaccia.scrivi("Un cadetto ha effettuato l'altro giorno delle riparazioni fuori dall'astronave");
    interfaccia.scrivi("Aveva, non so come, portato con se' la foto di sua moglie.");
    interfaccia.scrivi("Con suo grande dispiacere l'ha vista roteare nello spazio.");
    interfaccia.scrivi("Persa per sempre.");
}
}

```

L'azione 94 è relativa alla lettura del "file15" il quale contiene in maniera indiretta un utile suggerimento sul corretto utilizzo del compartimento stagno.

```

void Astro::azione_94(){
    system("cls");
    if(oggetti.get_oggetto(58).get_luogo() != 0)
    {
        interfaccia.scrivi("Puoi leggere i file solo dopo averli scaricati dal terminale dell'archivio");
    }
    else
    {
        interfaccia.scrivi("File15");
    }
}

```

```

interfaccia.scrivi("Ieri nella sala mensa e' scoppiata una rissa.");
interfaccia.scrivi("A quanto pare uno dei cadetti aveva prodotto delle osservazioni poco lusinghiere su un collega.");
interfaccia.scrivi("L'armonia dell'equipaggio e' fondamentale per portare a termine la missione.");
interfaccia.scrivi("Sono stati presi severi provvedimenti nei confronti dei fautori della rissa.");
interfaccia.scrivi("Simili accadimenti non dovranno ripetersi.");
}
}

```

L'azione 95 è relativa alla lettura del "file21" il quale contiene in maniera indiretta un utile suggerimento sul corretto utilizzo del reattore.

void Astro::azione_95(){

```

system("cls");
if(oggetti.get Oggetto(71).get_luogo() != 0)
{
    interfaccia.scrivi("Puoi leggere i file solo dopo averli scaricati dal terminale dell'archivio");
}
else
{
    interfaccia.scrivi("File21");
    interfaccia.scrivi("Il reattore dell'astronave ha fatto registrare dei valori anomali negli ultimi giorni.");
    interfaccia.scrivi("Ne ho parlato con il comandante e sembra d'accordo nell'avviare un'indagine approfondita.");
    interfaccia.scrivi("Dobbiamo essere operativi al cento per cento per proseguire.");
    interfaccia.scrivi("Stilero' un manuale da consultare in caso di emergenza.");
    interfaccia.scrivi("Mi sembra la cosa piu' giusta da fare.");
}

```

```
}
```

L'azione 96 permette al giocatore di poter entrare nelle gallerie del museo:

```
void Astro::azione_96(){

    system("cls");

    int num_persone=(rand()+1)%11;

    bool uscita_museo=false;

    bool uscita_coda=false;

    string risposta_attesa; //input per la risposta

    int risposta_documento; //input per la risposta al documento

    while (!uscita_coda)

    {

        if (num_persone==0)

        {

            uscita_coda=true;

        }

        else

        {

            interfaccia.scrivi_parziale("Ci sono ");

            interfaccia.scrivi_parziale(num_persone);

            interfaccia.scrivi(" persone in coda per pagare il biglietto /n");

            interfaccia.scrivi("Il costo del biglietto e' 10 euro");

            interfaccia.scrivi("Desideri aspettare e pagare? (s/n):");

            cin >> risposta_attesa;

            if (risposta_attesa=="s" || risposta_attesa=="sÃ¬" || risposta_attesa=="si")

            {   bool controlla=false;

                tempo=tempo-num_persone;
```

```

uscita_coda=true;
controlla=biglietto_museo();
if(controlla==false)
    uscita_museo=true;
}
else
if (risposta_attesa=="n" || risposta_attesa=="no")
{
    uscita_museo=true;
    uscita_coda=true;
}
else
{
    interfaccia.scrivi("Input non compreso. Digitare correttamente le
risposte!");
}
}
}

```

```

while (!uscita_museo)
{
// Menu di scelta 0,1,...
system("cls");

```

```

interfaccia.scrivi(" -----");
interfaccia.scrivi("|Benvenuto nel museo delle scienze
|");

```

```

interfaccia.scrivi("Hai a disposizione una serie di gallerie visitabili, ognuna
delle quali      |");
interfaccia.scrivi("|Presenta diversi percorsi con varie tipologie di esposizioni.
|");
interfaccia.scrivi("|Scegli il percorso che vuoi visitare, altrimenti premi 5 per
uscire:      |");
interfaccia.scrivi("|1 <- Galleria A      |");
interfaccia.scrivi("|2 <- Galleria B      |");
interfaccia.scrivi("|3 <- Galleria C      |");
interfaccia.scrivi("|4 <- Galleria D      |");
interfaccia.scrivi("|5 <- Uscire dal museo
|");
interfaccia.scrivi(" -----\n-----\n");
interfaccia.scrivi_parziale("Mosse rimaste: ");
interfaccia.scrivi_parziale(tempo);
interfaccia.scrivi_parziale("\n");
interfaccia.scrivi_parziale("Risposta: ");
cin >> risposta_documento;

switch (risposta_documento) {
    case 1:
        system("cls");
        mostra_galleria_a();

    break;
    case 2:
        system("cls");
        mostra_galleria_b();
    break;
}

```

case 3:

```
    system("cls");
    mostra_galleria_c();
    break;
```

case 4:

```
    system("cls");
    mostra_galleria_d();
    break;
```

case 5:

```
uscita_museo=true;
system("cls");
break;
```

default:

```
interfaccia.scrivi("Comando non riconosciuto");
uscita_museo=true;
system("pause");
```

}

}

5.3.4 Gioco.cpp

All'interno della classe Gioco.cpp sono state realizzate delle funzioni e delle procedure di supporto all'azione 96 inerenti al luogo "Museo":

- biglietto_museo
- mostra_galleria_a
- mostra_galleria_b
- mostra_galleria_c
- mostra_galleria_d

La funzione *biglietto_museo* è di tipo booleano, la sua funzione è quella di controllare se il giocatore sia in possesso del portafoglio o del credito disponibile per l'acquisto del biglietto.

In caso di esito negativo la funzione restituisce "false" e stampa il messaggio che richiede di prendere il

portafoglio o di raggiungere il credito necessario per l'acquisto(10€), altrimenti restituisce "true" e acquista il biglietto , con la possibilità di ricevere randomicamente un Bonus di 40 punti. Il bonus sarà gestito da una procedura (Aggiorna_Tempo_bonus()).

```
void Gioco::Aggiorna_Tempo_bonus() // Museo
{
    int tempo_b=40;
    interfaccia.scrivi("-----Complimenti hai ottenuto 40 punti bonus!-----
-----");
    cout<<"Tempo precedente: "<<tempo<<endl;
    tempo=tempo+tempo_b;
    cout<<"Il tempo attuale e':"<<tempo<<endl;

}
```

bool Gioco::biglietto_museo() //funzione ausiliare per controllare la disponibilità del portafoglio

```
// e acquistare biglietto museo
{
    bool acquisto=false;
    char a[1];
    int tempo_b=0;
    float saldo_p=0.0f;
    int numero;
    bool trovato = false;
    trovato = portafoglio.hai_Portafoglio(oggetti);
    if(trovato==true)
    {
        saldo_p=portafoglio.get_contanti();
        if (saldo_p>=10)
        {
            saldo_p=saldo_p-10;
```

```

portafoglio.set_contanti(saldo_p);

do

{
    system("cls");
    interfaccia.scrivi("Biglietto acquistato!");
    interfaccia.a_capo();
    //Calcolo bonus randomico
    numero=rand()%10+1;
    if(numero==3)
    {
        Aggiorna_Tempo_bonus();
    }
    interfaccia.scrivi("Premi un tasto per continuare...");
}

cin>>a;
} while(a==" ");
acquisto=true;
}

else{
    interfaccia.scrivi("Credito insufficiente!");
}
else if(trovato==false)
{
    interfaccia.scrivi("Non e' possibile acquistare se non indossi
il portafoglio con almeno 10 euro al suo interno! ");
}

```

```
return acquisto;
```

```
}
```

La procedura mostra_galleria_a permette di esplorare la galleria a e di poter interagire con la relativa guida elettronica, ossia visualizzare i reperti e chiederne la descrizione:

```
void Gioco::mostra_galleria_a()
```

```
{
```

```
int risposta_documento;
```

```
bool uscita_museo;
```

```
do
```

```
{
```

```
    system("cls");
```

```
    uscita_museo=false;
```

```
    interfaccia.scrivi("|-----  
----|");
```

```
    interfaccia.scrivi("|BENVENUTO NELLA GALLERIA A  
|");
```

```
    interfaccia.scrivi("|In questa galleria verrai guidato nell'esplorazione del percorso  
Preistorico. |");
```

```
    interfaccia.scrivi("|Il percorso Preistorico comprende una serie di esposizioni di  
notevole importanza.|");
```

```
    interfaccia.scrivi("|Per maggiori descrizioni sui reperti presenti, consultare la guida  
elettronica : |");
```

```

    interfaccia.scrivi("|1 - Neuropteris sp.-Felci fossili-Carbonifero-Francia
|");
    interfaccia.scrivi("|2 - Allosaurus fragilis-Scheletro di Allosauro-Giurassico-Stati
Uniti d'America |");
    interfaccia.scrivi("|3 - Coproliti di dinosauri - Epoche varie - Stati Uniti d'America
|");
    interfaccia.scrivi("|4 - Pterodactylus kochi-Rettile volante-Giurassico-Solnhofen
(Germania) |");
    interfaccia.scrivi("|5 - Tyrannosaurus rex-Palena (Chieti)
|");
    interfaccia.scrivi("|6 - PER USCIRE DALLA GUIDA ELETTRONICA
|");
    interfaccia.scrivi("-----|-----|-----|");
    interfaccia.scrivi_parziale("Mosse rimaste: ");
    tempo--;
    interfaccia.scrivi_parziale(tempo);
    interfaccia.scrivi_parziale("\n");
    interfaccia.scrivi_parziale("Risposta: ");

cin >> risposta_documento;

switch (risposta_documento) {
    case 1:
        system("cls");
        interfaccia.scrivi("-----|-----|-----|");
        interfaccia.scrivi("|Neuropteris sp.-Felci fossili-Carbonifero-Francia
|");

```

```
interfaccia.scrivi("|-----  
|");  
interfaccia.scrivi("|Queste piante raggiungevano un'altezza di cinque metri, con un  
tronco molto |");  
interfaccia.scrivi("|simile a quello delle attuali palme. Il fusto era formato in parte  
dalle |");  
interfaccia.scrivi("|basi foliari delle vecchie foglie, e possedeva radici aeree che  
crescevano |");  
interfaccia.scrivi("|in prossimitÃ della base. Le fronde erano di diverso tipo: alcune  
erano |");  
interfaccia.scrivi("|dentellate, altre erano munite di foglioline arrotondate. A molte  
di queste |");  
interfaccia.scrivi("|foglie sono stati assegnati nomi diversi, come Neuropteris o  
Alethopteris. |");  
interfaccia.scrivi("|La prima era un tipo di foglia di grandi dimensioni, con un  
rachide percorso|");  
interfaccia.scrivi("|da striature longitudinali. In Alethopteris, invece, le venature  
erano poco |");  
interfaccia.scrivi("|visibili e quasi ad angolo retto, mentre il margine della foglia era  
|");  
interfaccia.scrivi("|dentellato. |");  
interfaccia.scrivi("|-----  
|");  
tempo--;  
system("pause");  
break;  
case 2:  
    system("cls");  
    interfaccia.scrivi("|-----  
|");  
    interfaccia.scrivi("|Allosaurus fragilis-Scheletro di Allosauro-Giurassico-Stati Uniti  
d'America |");
```

```
interfaccia.scrivi("-----  
");  
interfaccia.scrivi("|Allosaurus e' un genere estinto di grande dinosauro teropode,  
vissuto tra i |");  
interfaccia.scrivi("|155 e i 145 milioni di anni fa, durante il periodo Giurassico. I  
primi |");  
interfaccia.scrivi("|resti fossili furono ritrovati nel 1877, ad opera del paleontologo  
Othniel |");  
interfaccia.scrivi("|Charles Marsh, che ribattezzÃ² i resti come Antrodemus.  
Essendo uno dei primi|");  
interfaccia.scrivi("|dinosauri teropodi meglio conservati e piu' completi, questo  
animale ha piu'|");  
interfaccia.scrivi("|volte attirato l'attenzione di paleontologi e amanti dei dinosauri.  
|");  
interfaccia.scrivi("|L'Allosaurus era un predatore bipede di modeste dimensioni; il  
suo cranio |");  
interfaccia.scrivi("|era incredibilmente robusto e compatto e armato di una  
moltitudine di denti.|");  
interfaccia.scrivi("|La lunghezza di un esemplare adulto non doveva essere inferiore  
agli 8,5 |");  
interfaccia.scrivi("|metri di lunghezza, anche se alcuni resti frammentari  
suggeriscono |");  
interfaccia.scrivi("|dimensioni maggiori, con esemplari che avrebbero potuto  
raggiungere i |");  
interfaccia.scrivi("|12 metri di lunghezza. |");  
interfaccia.scrivi("-----  
");  
tempo--;  
system("pause");  
break;  
case 3:  
system("cls");
```

```
interfaccia.scrivi("-----  
|");  
interfaccia.scrivi("|Coproliti di dinosauri - Epoche varie - Stati Uniti d'America  
|");  
interfaccia.scrivi("-----  
|");  
interfaccia.scrivi("|Il termine coprolite deriva dal greco kopros (sterco) e l'Athos  
(pietra) e |");  
interfaccia.scrivi("|indica un escremento, prodotto da un animale vissuto nel  
passato, che si e' |");  
interfaccia.scrivi("|fossilizzato. Dall'analisi di questi reperti, si possono ricavare  
|");  
interfaccia.scrivi("|informazioni sulle abitudini alimentari e l'habitat nel quale  
l'animale |");  
interfaccia.scrivi("|viveva. Per esempio, la presenza di semi, foglie, corteccia o  
radici indica |");  
interfaccia.scrivi("|che l'escremento e' stato prodotto da un erbivoro, mentre se  
si individuano |");  
interfaccia.scrivi("|frammenti di ossa, artigli o tendini l'animale era un  
carnivoro. |");  
interfaccia.scrivi("-----  
|");  
tempo--;  
system("pause");  
break;  
case 4:  
system("cls");  
interfaccia.scrivi("-----  
|");  
interfaccia.scrivi("|Pterodactylus kochi-Rettile volante-Giurassico-Solnhofen  
(Germania) |");
```

```
interfaccia.scrivi("-----  
|");  
interfaccia.scrivi("|Pterodactylus e' un estinto genere di pterosauro, i cui membri  
sono |");  
interfaccia.scrivi("|popolarmente chiamati "pterodattili". Attualmente, il genere  
contiene una |");  
interfaccia.scrivi("|singola specie, Pterodactylus antiquus, che oltre ad essere la  
specie tipo |");  
interfaccia.scrivi("|e' anche il primissimo genere di pterosauro mai rinvenuto. I  
principali |");  
interfaccia.scrivi("|ritrovamenti di resti fossili di questo animale sono stati rinvenuti  
|");  
interfaccia.scrivi("|principalmente, nei Calcari di Solnhofen, di Baviera, in  
Germania, risalenti|");  
interfaccia.scrivi("|alla fine del periodo Giurassico, circa 150,8-148-500 milioni di  
anni fa, |");  
interfaccia.scrivi("|anche se alcuni resti frammentari sono stati rinvenuti anche in  
altre aree |");  
interfaccia.scrivi("|in Europa e in Africa. Questo animale era un predatore che  
probabilmente si |");  
interfaccia.scrivi("|cibava soprattutto di pesci e piccoli invertebrati marini. Come  
tutti gli |");  
interfaccia.scrivi("|pterosauri, anche le ali dello Pterodactyluserano formate da una  
membrana |");  
interfaccia.scrivi("|di pelle che si estendeva dalla fine del quarto dito della "mano"  
FINO |");  
interfaccia.scrivi("|agli arti posteriori. L'ala era supportata, ulteriormente,  
internamente da |");  
interfaccia.scrivi("|fibre di collagene ed esternamente da strutture cheratinose.  
|");  
interfaccia.scrivi("-----  
|");  
tempo--;
```

```

system("pause");
break;
case 5:
system("cls");
interfaccia.scrivi("-----");
interfaccia.scrivi("Pterodactylus kochi-Rettile volante-Giurassico-Solnhofen  

(Germania) -----");
interfaccia.scrivi("-----");
interfaccia.scrivi("Il tirannosauro era un dinosauro vissuto nel Cretaceo superiore  

-----");
interfaccia.scrivi("|appartenente alla famiglia dei tirannosauridi. Visse in  

nordamerica, che -----");
interfaccia.scrivi("|anticamente era un continente isolato nominato Laramidia. Il  

Tyrannosaurus -----");
interfaccia.scrivi("|era molto piu' diffuso geograficamente degli altri tirannosauridi.  

I suoi -----");
interfaccia.scrivi("|fossili si trovano in una varietÃ di formazioni risalenti all' epoca  

-----");
interfaccia.scrivi("|Maastrichtiana del Cretaceo superiore, circa 68-66 milioni di  

anni fa. -----");
interfaccia.scrivi("|Fu una delle specie degli ultimi dinosauri non-aviani viventi  

quando si ebbe|");
interfaccia.scrivi("|l' estinzione di massa del Cretaceo-Paleocene, che determino' la  

scomparsa |");
interfaccia.scrivi("|dei dinosauri propriamente detti. -----");
interfaccia.scrivi("-----");
tempo--;
system("pause");
break;

```

```

case 6:
    uscita_museo=true;
    system("cls");
    break;
default:
    interfaccia.scrivi("Comando non riconosciuto");

    system("pause");
    break;

}
} while(uscita_museo!=true);
}

```

La stessa procedura vale per la mostra delle gallerie b,c,d richiamate all'interno dello switch dell'azione 96:

void Gioco::mostra_galleria_b()

{ **bool** uscita_museo;

do

{

uscita_museo=**false**;

system("cls");

int risposta_documento;

interfaccia.scrivi("|-----
-----|");

interfaccia.scrivi("|BENVENUTO NELLA GALLERIA B
|");

```
    interfaccia.scrivi("|In questa galleria verrai guidato nell'esplorazione del percorso  
Romano. |");  
  
    interfaccia.scrivi("|Come testimonianza di questo periodo storico potrete ammirare  
antichi affreschi |");  
  
    interfaccia.scrivi("|Per ulteriori dettagli, consultare la guida elettronica:  
|");  
  
    interfaccia.scrivi("|1 - Numa Pompilio istituisce il culto delle Vestali e dei  
sacerdoti |");  
  
    interfaccia.scrivi("|2 - Ritrovamento della Lupa con Romolo e Remo  
|");  
  
    interfaccia.scrivi("|3 - Combattimento degli Orazi e Curiazi  
|");  
  
    interfaccia.scrivi("|4 - Ratto delle Sabine  
|");  
  
    interfaccia.scrivi("|5 - Battaglia di Tullo Ostilio contro i Veienti e i Fidenati  
|");  
  
    interfaccia.scrivi("|6 - PER USCIRE DALLA GUIDA ELETTRONICA  
|");  
  
    interfaccia.scrivi("-----  
-----|");  
  
    interfaccia.scrivi_parziale("Mosse rimaste: ");  
  
    interfaccia.scrivi_parziale(tempo);  
  
    interfaccia.scrivi_parziale("\n");  
  
    interfaccia.scrivi_parziale("Risposta: ");  
  
  
    cin >> risposta_documento;  
  
    switch (risposta_documento) {  
  
        case 1:  
  
            system("cls");  
  
            interfaccia.scrivi("-----  
|");
```

```
    interfaccia.scrivi("|Numa Pompilio istituisce il culto delle Vestali e dei sacerdoti(1636-1638) |");  
    interfaccia.scrivi("|-----|");  
    interfaccia.scrivi("|Al centro della scena, sullo sfondo di un grandioso scorci architettonico, |");  
    interfaccia.scrivi("|arde sull' altare il fuoco sacro che le Vestali dovevano custodire sempre |");  
    interfaccia.scrivi("|acceso. |");  
    interfaccia.scrivi("|-----|");  
    tempo--;  
    system("pause");  
break;  
case 2:  
    system("cls");  
    interfaccia.scrivi("|-----|");  
    interfaccia.scrivi("|Ritrovamento della Lupa con Romolo e Remo \(1596\) |");  
    interfaccia.scrivi("|-----|");  
    interfaccia.scrivi("|Faustolo scopre sotto i rami di un fico, sulla riva del Tevere, la Lupa che |");  
    interfaccia.scrivi("|allatta Romolo e Remo. Nella figura della lupa e' evidente il richiamo alla |");  
    interfaccia.scrivi("|Lupa capitolina conservata nel palazzo e simbolo della città . |");  
    interfaccia.scrivi("|-----|");  
    tempo--;  
    system("pause");
```



```
    interfaccia.scrivi("|In primo piano e' il gruppo delle donne Sabine rapite dai  
Romani per |");  
    interfaccia.scrivi("|popolare la cittÃ da poco fondata. L'affresco, eseguito dopo  
circa venti |");  
    interfaccia.scrivi("|anni di interruzione, condivide con le ultime due scene una  
tecnica |");  
    interfaccia.scrivi("|pittorica piu' rapida e sommaria, tipica della tarda maniera del  
|");  
    interfaccia.scrivi("|Cavalier d' Arpino. |");  
    interfaccia.scrivi("|-----|");  
    tempo--;  
    system("pause");  
    break;  
    case 5:  
        system("cls");  
        interfaccia.scrivi("|-----|");  
        interfaccia.scrivi("|Battaglia di Tullo Ostilio contro i Veienti e i Fidenati(1597-  
1601) |");  
        interfaccia.scrivi("|-----|");  
        interfaccia.scrivi("|Con vivacita' e' rappresentato un episodio della guerra di  
espansione |");  
        interfaccia.scrivi("|intrapresa dai Romani contro le cittÃ vicine al tempo di Tullo  
Ostilio, |");  
        interfaccia.scrivi("|terzo re di Roma. |");  
        interfaccia.scrivi("|-----|");  
        tempo--;  
        system("pause");
```

```
break;  
case 6:  
uscita_museo=true;  
system("cls");  
break;  
default:  
interfaccia.scrivi("Comando non riconosciuto");  
system("pause");  
}  
  
}  
  
}
```

```
void Gioco::mostra_galleria_c()  
{bool uscita_museo;  
  
do  
{  
uscita_museo=false;  
system("cls");  
int risposta_documento;  
interfaccia.scrivi("|-----  
----|");  
interfaccia.scrivi("|BENVENUTO NELLA GALLERIA C  
|");  
interfaccia.scrivi("|Questa galleria comprende i ritrovamenti del periodo che va dal  
200-300 a.C. |");
```

```
    interfaccia.scrivi("|\u00c9 sono stati ritrovati numerosi reperti risalenti a questa epoca,  
    alcuni dei quali |");
```

```
    interfaccia.scrivi("|\u00c9 sono presenti all'interno della nostra galleria.  
    |");
```

```
    interfaccia.scrivi("|\u00c9 Per maggiori descrizioni, consultare la guida elettronica:  
    |");
```

```
    interfaccia.scrivi("|\u00c9 1 - Stele funeraria del tipo \u00e0\u00e0a falsa porta\u2665 a nome di  
    Sameri |");
```

```
    interfaccia.scrivi("|\u00c9 2 - Rilievo con scena di libagione  
    |");
```

```
    interfaccia.scrivi("|\u00c9 3 - Rilievo dalla tomba di Horemheb con scena di lavoro nei  
    campi dell'\u2122oltretomba|");
```

```
    interfaccia.scrivi("|\u00c9 4 - Rilievo con la dea Renenutet  
    |");
```

```
    interfaccia.scrivi("|\u00c9 5 - Sarcofago a cassa a nome di Irinimenpu  
    |");
```

```
    interfaccia.scrivi("|\u00c9 6 - PER USCIRE DALLA GUIDA ELETTRONICA  
    |");
```

```
    interfaccia.scrivi("|\u2014\u2014\u2014|\u2014\u2014\u2014");
```

```
    interfaccia.scrivi_parziale("Mosse rimaste: ");
```

```
    interfaccia.scrivi_parziale(tempo);
```

```
    interfaccia.scrivi_parziale("\n");
```

```
    interfaccia.scrivi_parziale("Risposta: ");
```

```
    cin >> risposta_documento;
```

```
    switch (risposta_documento) {
```

```
        case 1:
```

```
            system("cls");
```

```
            interfaccia.scrivi("|\u2014\u2014\u2014|\u2014\u2014\u2014");
```

```
            interfaccia.scrivi("|\u00c9 Stele funeraria del tipo a \"falsa porta\" a nome di Sameri  
            |");
```

interfaccia.scrivi("-----");

interfaccia.scrivi("|La stele a " falsa porta " e' un elemento funerario tipico delle |");

interfaccia.scrivi("|sepolture dell' Antico Regno ed e' costituita da due o piu'
montanti |");

interfaccia.scrivi("|laterali e da un architrave sotto cui e' scolpita una stuoia
arrotolata |");

interfaccia.scrivi("|a suggerire l' idea di una porta accessibile all' anima del
defunto. Questa |");

interfaccia.scrivi("|stele appartiene ad un funzionario di nome Sameri, membro di
una famiglia di |");

interfaccia.scrivi("|cortigiani molto vicina al faraone, e gli dedicata dal padre
Urkapkah, che |");

interfaccia.scrivi("|include nel dono anche la moglie e gli altri figli. Sameri e'
rappresentato |");

interfaccia.scrivi("|con la tecnica del rilievo ad incavo sull' architrave della
falsa porta, |");

interfaccia.scrivi("|mentre siede in compagnia della madre Henutes davanti a una
tavola ricolma |");

interfaccia.scrivi("|di vasellame da mensa e di alimenti (pani, anatre giÃ spennate,
tranci di |");

interfaccia.scrivi("|carne bovina, etc.) utili per la sua sopravvivenza ultraterrena
che |");

interfaccia.scrivi("|â€™iscrizione in caratteri geroglifici sottostante completa per
forza magica |");

interfaccia.scrivi("|Lâ€™importanza del pane, quale alimento base della dieta
egiziana, appare |");

interfaccia.scrivi("|evidente, anche se non e' possibile differenziare le varie
tipologie di |");

interfaccia.scrivi("|pane elencate per qualitÃ di ingredienti, modalitÃ di impasto e
di cottura |");

```
interfaccia.scrivi("-----  
");  
  
tempo--;  
system("pause");  
break;  
  
case 2:  
  
    system("cls");  
  
    interfaccia.scrivi("-----  
");  
  
    interfaccia.scrivi("Rilievo con scena di libagione ");  
  
    interfaccia.scrivi("-----  
");  
  
    interfaccia.scrivi("Il rilievo, collocato in origine alle pareti di una tomba, mostra  
una scena |");  
  
    interfaccia.scrivi("di libagione. La signora della casa Nubi, proprietaria  
della sepoltura, |");  
  
    interfaccia.scrivi("siede su una bassa sedia decorata con zampe di gatto mentre  
riceve l'offerta|");  
  
    interfaccia.scrivi("funeraria di acqua e di vino da sua figlia Kiki, in piedi davanti a  
lei. |");  
  
    interfaccia.scrivi("Altri cibi, tra i quali un mazzetto di porri e varie cucurbitacee,  
sono |");  
  
    interfaccia.scrivi("raccolti all'interno di un profondo bacile alle spalle di Kiki  
allo scopo di|");  
  
    interfaccia.scrivi("nutrire in eterno la defunta. L'eleganza e la morbidezza delle  
figure, come |");  
  
    interfaccia.scrivi("la raffinatezza dei costumi rivelano l'agiatezza di vita della  
dama Nubi, |");  
  
    interfaccia.scrivi("tipica dell' Egitto del Nuovo Regno. Entrambe le donne  
indossano lunghi abiti|");
```

```
    interfaccia.scrivi("la cui semplicitÃ  contrasta con la resa accurata dei gioielli,  
visibili alle");  
  
    interfaccia.scrivi("braccia, al collo e alle orecchie,e delle pesanti parrucche adorne  
di nastri");  
  
    interfaccia.scrivi("|coroncine di fiori e cono di unguento profumato.  
|");  
  
    interfaccia.scrivi("|-----  
|");  
  
    tempo--;  
  
    system("pause");  
  
break;  
  
case 3:  
  
    system("cls");  
  
    interfaccia.scrivi("|-----  
-|");  
  
    interfaccia.scrivi("Rilievo dalla tomba di Horemheb con scena di lavoro nei campi  
dell'oltretomba |");  
  
    interfaccia.scrivi("|-----  
-|");  
  
    interfaccia.scrivi("Il rilievo, proveniente da una delle tre cappelle di culto annesse  
alla tomba |");  
  
    interfaccia.scrivi("menfita del generale Horemheb, e' costituito da due frammenti  
parietali |");  
  
    interfaccia.scrivi("combacianti ed e' suddiviso in quattro fasce orizzontali scolpite  
a |");  
  
    interfaccia.scrivi("bassorilievo. È probabile che quella in alto, di cui sopravvive  
solo la |");  
  
    interfaccia.scrivi("parte inferiore, contenesse l'omaggio di Horemheb alle  
divinitÃ  funerarie. |");  
  
    interfaccia.scrivi("Nei due registri centrali, meglio conservati, Horemheb e' colto  
in momenti |");
```

```
    interfaccia.scrivi("|diversi: seduto in compagnia della sua anima akh dalle  
    simbianze di uccello |");
```

```
    interfaccia.scrivi("|appollaiato su un trespolo davanti a una tavola piena di pani di  
    varia forma, |");
```

```
    interfaccia.scrivi("|tranci di carne bovina e altri cibi destinati al pasto funebre del  
    defunto; |");
```

```
    interfaccia.scrivi("|in piedi mentre governa alcuni buoi che calpestano un mucchio  
    di spighe di |");
```

```
    interfaccia.scrivi("|cereale per separare i chicchi dalla pula; in piedi, dietro un  
    aratro trainato|");
```

```
    interfaccia.scrivi("|da due buoi, intento a dissodare il terreno da coltivare. In basso,  
    Horemheb |");
```

```
    interfaccia.scrivi("|e' nuovamente seduto davanti a una tavola stracolma di offerte  
    e, a raccolto |");
```

```
    interfaccia.scrivi("|ultimato, riceve in dono alcuni mannelli di lino da parte di tre  
    contadini. |");
```

```
    interfaccia.scrivi("|Tutte queste scene, ispirate al capitolo 110 del Libro dei Morti,  
    ci mostrano |");
```

```
    interfaccia.scrivi("|Horemheb al lavoro nei campi dell'oltretomba per garantirsi  
    la sopravvivenza |");
```

```
    interfaccia.scrivi("|eterna. Il serpente ureo, simbolo di regalitÃ , che orna la sua  
    fronte, fu |");
```

```
    interfaccia.scrivi("|scalpellato in un secondo momento, solo quando Horemheb  
    divenne faraone |");
```

```
    interfaccia.scrivi("|d'Egitto alla fine della XVIII dinastia.  
    |");
```

```
    interfaccia.scrivi("-----  
-|");
```

```
    tempo--;
```

```
    system("pause");
```

```
break;
```

case 4:

```
system("cls");
interfaccia.scrivi("|-----|");
interfaccia.scrivi("|Rilievo con la dea Renenutet |");
interfaccia.scrivi("|-----|");
interfaccia.scrivi("|Il rilievo proviene dalla tomba di uno scriba di nome Amenemhat, vissuto agli |");
interfaccia.scrivi("|inizi del Nuovo Regno, prima del faraone Akhenaton, durante il cui governo il |");
interfaccia.scrivi("|nome del dio Amon fu spesso cancellato, come in questo caso. Sul rilievo sono |");
interfaccia.scrivi("|raffigurate due tipiche attività agricole: nella parte alta, alcuni contadini |");
interfaccia.scrivi("|puliscono il grano lanciandolo in aria con sessole di legno per separare i |");
interfaccia.scrivi("|chicchi della pula, mentre un uomo offre loro da bere con profonde ciotole |");
interfaccia.scrivi("|emisferiche, riempite in un grande vaso panciuto alle sue spalle. Nel registro|");
interfaccia.scrivi("|inferiore si procede alla misurazione del cereale ormai pulito e accumulato di|");
interfaccia.scrivi("|fronte alla dea delle messi Renenutet, in forma di serpente cobra. Scene di |");
interfaccia.scrivi("|questo tipo, sia dipinte che a rilievo, sono piuttosto comuni nell'arte |");
interfaccia.scrivi("|egiziana a partire dall'Antico Regno |");
interfaccia.scrivi("|-----|");
tempo--;
```

```
    system("pause");

break;

case 5:

    system("cls");

    interfaccia.scrivi("|-----|");
    interfaccia.scrivi("|Sarcofago a cassa a nome di Irinimenpu|");
    interfaccia.scrivi("|-----|");
    interfaccia.scrivi("|La decorazione principale di questo sarcofago destinato al corredo funerario |");
    interfaccia.scrivi("|di un egiziano di nome Irinimenpu consiste in una serie di pannelli a |");
    interfaccia.scrivi("|'facciata di palazzo', un motivo ripreso dall'architettura funararia |");
    interfaccia.scrivi("|dell'Antico Regno, che rende il sarcofago simile a una dimora eterna. Su uno |");
    interfaccia.scrivi("|dei lati lunghi della cassa, in posizione centrale, sono raffigurate numerose |");
    interfaccia.scrivi("|offerte alimentari (pani, ortaggi, frutti, tranci di carni bovine, volatili |");
    interfaccia.scrivi("|giÃ spennati, contenitori per liquidi, etc.), perche' il defunto possa |");
    interfaccia.scrivi("|nutrirsiene per forza magica nella vita ultraterrena. Sullo stesso lato, |");
    interfaccia.scrivi("|a una delle estremita', e' dipinta una porta chiusa, sopra la quale sono |");
    interfaccia.scrivi("|collocati due occhi che indicano la presenza all'interno della testa della |");
```

```

    interfaccia.scrivi("mummia e allo stesso tempo rappresentano una protezione
magica per il corpo |");
    interfaccia.scrivi("del defunto. L'ottimo stato di conservazione del legno e dei
vivaci colori |");
    interfaccia.scrivi("utilizzati per decorarlo si deve al clima caldo e asciutto
dell'Egitto, |");
    interfaccia.scrivi("oltre che alla collocazione originaria del sarcofago
nell'ambiente protetto |");
    interfaccia.scrivi("della sepoltura. |");
    interfaccia.scrivi("-----|");
    tempo--;
    system("pause");

break;
case 6:
uscita_museo=true;
system("cls");
break;
default:
    interfaccia.scrivi("Comando non riconosciuto");
    system("pause");
}

}while(uscita_museo!=true);
}

void Gioco::mostra_galleria_d()
{
bool uscita_museo;

do
{

```

```
uscita_museo=false;

system("cls");
int risposta_documento;

interfaccia.scrivi("|-----|");
interfaccia.scrivi("|BENVENUTO NELLA GALLERIA D |");
interfaccia.scrivi("|In questa galleria verrai guidato nell'esplorazione del percorso sulla Magna Grecia.|");
interfaccia.scrivi("|La Magna Grecia e' l'area geografica della penisola italiana meridionale che fu |");
interfaccia.scrivi("|anticamente colonizzata dai Greci a partire dall' VIII secolo a.C.. |");
interfaccia.scrivi("|Di questa civiltà possediamo numerose sculture e reperti |");
interfaccia.scrivi("|Per maggiori descrizioni sui reperti presenti, consultare la guida elettronica: |");
interfaccia.scrivi("|1 - Testa in marmo di Eracle |");
interfaccia.scrivi("|2 - Stele figurata in marmo con defunto in nudità eroica. |");
interfaccia.scrivi("|3 - Statua marmorea del II secolo d.C. di genio carpoftoro (portatore di frutti). |");
interfaccia.scrivi("|4 - Specchio in argento, con doratura a caldo. |");
interfaccia.scrivi("|5 - I Bronzi di Riace |");
interfaccia.scrivi("|6 - PER USCIRE DALLA GUIDA ELETTRONICA |");
```

```

interfaccia.scrivi("-----|");
-----|");

interfaccia.scrivi_parziale("Mosse rimaste: ");
interfaccia.scrivi_parziale(tempo);

interfaccia.scrivi_parziale("\n");
interfaccia.scrivi_parziale("Risposta: ");

cin >> risposta_documento;

switch (risposta_documento) {

case 1:

system("cls");

interfaccia.scrivi("-----|");
-----|");

interfaccia.scrivi("|Testa in marmo di Eracle |");
interfaccia.scrivi("-----|");
-----|");

interfaccia.scrivi("|Probabilmente una copia del colosso bronzeo creato da Lisippo
a Tara alla fine|");

interfaccia.scrivi("|del IV secolo AC. |");
interfaccia.scrivi("-----|");
-----|");

tempo--;
system("pause");

break;

case 2:

system("cls");

interfaccia.scrivi("-----|");
-----|");

interfaccia.scrivi("|Stele figurata in marmo con defunto in nuditÃ eroica. |");

```

```
interfaccia.scrivi("-----  
-|");  
  
interfaccia.scrivi("Raffigurato nell'atto di offrire una melagrana ad un  
serpente, simbolo      |");  
  
interfaccia.scrivi("funerario delle divinitÀ infernali.      |");  
  
interfaccia.scrivi("-----  
-|");  
  
tempo--;  
  
system("pause");  
  
break;  
  
case 3:  
  
system("cls");  
  
interfaccia.scrivi("-----  
-|");  
  
interfaccia.scrivi("Statua marmorea del II secolo d.C. di genio carpofo (portatore  
di frutti).      |");  
  
interfaccia.scrivi("-----  
-|");  
  
interfaccia.scrivi("Proveniente dall'area delle Terme e relativo alla decorazione  
scultorea delle      |");  
  
interfaccia.scrivi("stesse.      |");  
  
interfaccia.scrivi("-----  
-|");  
  
tempo--;  
  
system("pause");  
  
break;  
  
case 4:  
  
svstem("cls");
```

```
    interfaccia.scrivi("-----  
-|");  
    interfaccia.scrivi("||Specchio in argento, con doratura a caldo. |");  
    interfaccia.scrivi("-----  
-|");  
    interfaccia.scrivi("||È raffigurata l'immagine di una Musa, o di Afrodite, circondata  
da Eroti. |");  
    interfaccia.scrivi("||Dalla tomba degli Ori di Canosa.  
|");  
    interfaccia.scrivi("-----  
-|");  
    tempo--;  
    system("pause");
```

break;

case 5:

```
    system("cls");
```

```
    interfaccia.scrivi("-----  
-|");
```

```
    interfaccia.scrivi("||I Bronzi di Riace |");
```

```
    interfaccia.scrivi("-----  
-|");
```

```
    interfaccia.scrivi("||I Bronzi di Riace sono due statue di bronzo di provenienza  
greca o magnogreca |");
```

```
    interfaccia.scrivi("||o siceliota, databili al V secolo a.C. pervenute in eccezionale  
stato di |");
```

```
    interfaccia.scrivi("||conservazione. Le due statue, rinvenute il 16 agosto 1972 nei  
pressi di Riace,|");
```

```
    interfaccia.scrivi("||in provincia di Reggio Calabria sono considerate tra i  
capolavori scultorei |");
```

```
    interfaccia.scrivi("||piu' significativi dell'arte greca, e tra le testimonianze dirette dei  
grandi |");
```

```
interfaccia.scrivi("maestri scultori dell'eta' classica. Le ipotesi sulla provenienza e  
sugli    |");  
  
interfaccia.scrivi("autori delle statue sono diverse, ma non esistono ancora  
elementi che    |");  
  
interfaccia.scrivi("permettano di attribuire con certezza le opere ad uno specifico  
scultore.    |");  
  
interfaccia.scrivi("I Bronzi si trovano al Museo nazionale della Magna Grecia di  
Reggio Calabria, |");  
  
interfaccia.scrivi("luogo in cui sono stati riportati il 12 dicembre 2015, dopo la  
rimozione e il |");  
  
interfaccia.scrivi("soggiorno per tre anni (con annessi lavori di restauro) presso  
Palazzo    |");  
  
interfaccia.scrivi("Campanella, sede del Consiglio Regionale della Calabria a  
causa dei lavori di |");  
  
interfaccia.scrivi("ristrutturazione dello stesso museo. I Bronzi sono diventati uno  
dei simboli |");  
  
interfaccia.scrivi("della cittÃ stessa.    |");  
  
interfaccia.scrivi("-----  
-|");  
  
tempo--;  
  
system("pause");
```

break;

case 6:

uscita museo=true;

```
system("cls");
```

break;

default:

```
interfaccia.scrivi("Comando non riconosciuto");
```

```
system("pause");
```

}

5.3.5 Mappa.nav

Per poter aggiungere il luogo “archivio” ed il luogo “museo”, è stato necessario modificare il numero dei luoghi da 25 a 27, aggiornando le righe di navigazione per permettere al luogo di essere raggiunto. L’ubicazione dell’archivio è stata posta a sud del luogo “Aula” mentre il museo è stato posto a sud dell’ “Archivio”.

Per poter aggiungere il luogo denominato Archivio sono state modificate alcune righe e in particolare quelle in grado di permettere al luogo di essere raggiunto. Essendo stata stabilita la sua ubicazione a sud dell’Aula, è stata modificata prima di tutto la linea relativa all’Aula:

25,Nell’Aula,042600000000,via 1,1,1

Inoltre è stato creato il ventiseiesimo luogo all’interno dell’astronave denominato “Archivio”. È stata quindi aggiunta la seguente riga:

26,Nell’Archivio,252700000000,via 1,2,2

Sono infine state aggiunte le righe relative alle vie percorribili per raggiungere ed abbandonare il luogo Archivio.

25,26, via 1,sud,2,2,1,1

26,25, via 1,nord,2,2,1,1

Per quanto riguarda l’ubicazione del museo posto a sud dell’archivio:

27,Nel Museo,260000000000,via 1,2,2

Sono state aggiunte, infine, le righe relative alle vie percorribili per raggiungere ed abbandonare il luogo museo:

26,27 via 1,sud,2,2,1,1

27,26, via 1,nord,2,2,1,1

Inoltre la numerazione originale dell’archivio e del museo è stata cambiata in fase di integrazione in quanto il codice 24 era stato usato per l’Ufficio Postale, e il codice 25 per l’Aula .

Si è scelto quindi di utilizzare il codice 26 per l’archivio ed il codice 27 per il museo.

5.3.6 Descrizioni

Nella cartella “Descrizioni” è stato aggiunto il file di testo relativo all’Archivio e del Museo, modificando le descrizioni originali del progetto da integrare in modo da dare delle informazioni più precise sul luogo.

La descrizione dell’archivio (luogo 26) è la seguente:

Nell’Archivio. Vedo un computer

entrato nell’Archivio. Vedo un computer

Nuovamente nell’Archivio. Vedo un computer

Ancora una volta nell’Archivio. Vedo un computer

La descrizione del museo (luogo 27) è la seguente:

Nella biglietteria del Museo. Vedo 4 gallerie. Prova a entrare

entrato nella biglietteria del Museo. Vedo delle gallerie. Prova a entrare

nuovamente nella biglietteria del Museo. Vedo 4 gallerie. Prova a entrare

ritornato nella biglietteria del museo. Vedo delle gallerie. Prova a entrare

5.4 AULA SINGOLA

Sono state effettuate modifiche nei seguenti file:

- Astro.cpp
- Astro.h
- Mappa.nav

5.4.1 AGGIORNAMENTO DEI VOCABOLI DEL DIZIONARIO

Verranno inseriti nel dizionario i nuovi termini che saranno utili al riconoscimento dei nuovi comandi.

In Astro.cpp sono stati aggiunti i vocaboli degli oggetti e delle azioni mancanti.

//INIZIO modifiche Moschetti

```
vocabolario.inserisci("sedia", 92);
vocabolario.inserisci("accendi", 86);
vocabolario.inserisci("proiettore", 80);
vocabolario.inserisci("computer", 78);
vocabolario.inserisci("spegni", 72);
vocabolario.inserisci("lezione", 71);
vocabolario.inserisci("libri", 55);
vocabolario.inserisci("segui", 10);
```

//FINE modifiche Moschetti

5.4.2 Aggiornamento del dizionario degli oggetti

Verranno inseriti nell'apposito dizionario i seguenti oggetti.

In Astro.cpp sono stati aggiunti gli oggetti dell'aula.

//INIZIO modifiche Moschetti

```
oggetti.inserisci(Oggetto("un proiettore",80 , -25));
oggetti.inserisci(Oggetto("un computer",78 , -25));
oggetti.inserisci(Oggetto("dei libri",55 , 25));
oggetti.inserisci(Oggetto("delle sedie",92 , 25));
oggetti.inserisci(Oggetto("una lezione in corso",71 , -25));
```

//FINE modifiche Moschetti

5.4.3 Aggiornamento delle azioni di gioco

Verranno inserite le seguenti azioni

In Astro.cpp sono state aggiunte le seguenti azioni.

//INIZIO modifiche Moschetti

```
azioni.inserisci(249400, 85); // "Siediti"  
azioni.inserisci(249447, 85); // "Siediti sedia"  
azioni.inserisci(242558, 86); // "Leggi libri"  
azioni.inserisci(242944, 87); // "Usa computer"  
azioni.inserisci(244849, 88); // "Segui lezione"  
azioni.inserisci(244557, 89); // "Accendi proiettore"  
azioni.inserisci(244657, 90); // "Spegni proiettore"  
  
//FINE modifiche Moschetti.
```

In Astro.h sono state aggiunte le seguenti azioni.

//INIZIO modifiche Moschetti

```
void azione_85(); // Siediti per riposarti  
void azione_86(); // Leggi i libri presenti nell'aula  
void azione_87(); // Fare delle ricerche al computer  
void azione_88(); // Seguire la lezione  
void azione_89(); // Accendi il proiettore  
void azione_90(); // Spegni il proiettore
```

//FINE modifiche Moschetti

In Astro.cpp sono state aggiunte le seguenti azioni.

//INIZIO modifiche Moschetti

case 85 :

```
    azione_85();
```

```
    break;
```

case 86 :

```
    azione_86();
```

```
    break;
```

case 87 :

```

azione_87 () ;

break ;

case 88 :

    azione_88 () ;

    break ;

case 89 :

    azione_89 () ;

    break ;

case 90 :

    azione_90 () ;

    break ;

//FINE modifiche Moschetti

```

5.4.3.1 *Implementazione azione_85*

```

void Astro::azione_85(){

interfaccia.scrivi("Ti sei seduto");

storia_gioco.insStoria(stringa_comando , "Hai deciso di sederti"); //Aggiorna storia
gioco

}

```

5.4.3.2 *Implementazione azione_86*

```

bool controllo1=false;
bool controllo2=false;
bool controllo3=false;
Lista <int> l;
Lista <int>::posizione p;

```

void Astro::azione_86()

```

p=l.primolista();

```

```

int lib=(rand() % 3) + 1; // creo una variabile che mi randomizza un numero
da 1 a 3 che servirà per decidere quale libro leggere

if (lib==1 && controllo1==true){// se il libro e' stato già letto

    while(l.finelista(p)!=true){//fino a quando non raggiungiamo finelista

        if(l.leggilista(p)==1){

            interfaccia.scrivi("È UN LIBRO DI MANZONI");
            interfaccia.scrivi("L'HAI GIA' LETTO!!");

        }

        p=l.succlista(p);

    }

}

else if (lib==1 && controllo1==false) { //se il libro non è stato ancora letto

    interfaccia.scrivi("È UN LIBRO DI ALLESSANDRO MANZONI");
    interfaccia.scrivi("");
    interfaccia.scrivi("Ei fu. Sicome immobile,");
    interfaccia.scrivi("dato il mortal sospiro,");
    interfaccia.scrivi("stette la spoglia immemore ");
    interfaccia.scrivi("orba di tanto spiro,...");

l.inslista(1,p); // inserisce nella lista il numero 1
p=l.succlista(p);
controllo1=true; //metto a true il controllo della lettura del libro
}

```

int salute=Salute.GetStatoSalute(); // creo una variabile salute in cui metterò il valore della salute del giocatore

if(salute>=95){// se la salute è maggiore o uguale di 95 e non mai stato utilizzato il kit medico

```

    interfaccia.scrivi("");
}

```

```

        interfaccia.scrivi("HAI TROVATO UN KIT MEDICO MA LA
TUA SALUTE È ANCORA OTTIMA ");

        interfaccia.scrivi("LO USI MA NON HA NESSUN EFFETTO");
    }

else { // se la salute non è più maggiore di 95 e non è mai stato utilizzato
il kit medico

    interfaccia.scrivi("");
    Salute.SetStatoSalute(salute+5); // aggiorno la salute aggiungendo 5
unità

    interfaccia.scrivi("INCREDIBILE!!!");
    interfaccia.scrivi("HAI RECUPERATO CINQUE UNITÀ DI
SALUTE!!!");
    interfaccia.scrivi("ORA LA TUA SALUTE È DI:");
    cout << Salute.GetStatoSalute() << endl; // stampa a video della
salute aggiornata del giocatore

}

else if (lib==2 && controllo2==true){ // se il libro è stato già letto

while(l.finelista(p)!=true){ // fino a quando non raggiungiamo finelista
    if(l.leggilista(p)==2){

        interfaccia.scrivi("È UN LIBRO DI UNGARETTI");
        interfaccia.scrivi("L'HAI GIA' LETTO!!!");
    }

    p=l.succlista(p);
}

else if (lib==2 && controllo2==false) { // se il libro non è stato ancora letto
    interfaccia.scrivi("È UN LIBRO DI GIUSEPPE UNGARETTI");
    interfaccia.scrivi("");
}

```

```

interfaccia.scrivi("M'illumino d'immenso");

l.inslista(2,p); // inserisce nella lista il numero 2
p=l.succlista(p);
controllo2=true; //metto a true il controllo della lettura del libro
}

else if(lib==3 && controllo3==true){ // se il libro e' stato gia' letto

while(l.finlista(p)!=true){ // fino a quando non raggiungiamo finlista

if(l.leggilista(p)==3){

    interfaccia.scrivi("È UN LIBRO DI LEOPARDI");
    interfaccia.scrivi("L'HAI GIA' LETTO!!");

}

p=l.succlista(p);

}

}

else if (lib==3 && controllo3==false) { // se il libro non è stato ancora letto

    interfaccia.scrivi("È UN LIBRO DI GIACOMO LEOPARDI");
    interfaccia.scrivi("");
    interfaccia.scrivi("Silvia, rimembri ancora");
    interfaccia.scrivi("quel tempo della tua vita mortale,");
    interfaccia.scrivi("quando belta' splendea...");



l.inslista(3,p); // inserisce nella lista il numero 3
p=l.succlista(p);
controllo3=true; //metto a true il controllo della lettura del libro
}

```

```
storia_gioco.insStoria(stringa_comando , "hai deciso di leggere dei libri"); //Aggiorna storia gioco
}
```

5.4.3.3 *Implementazione azione_87*

```
void Astro::azione_87(){
```

```
    int num=(rand() % 4) + 1;// creo una variabile che mi randomizza un numero da 1 a 4 che servirà per decidere quale ricerca effettuare

    if (num==1){

        interfaccia.scrivi("Ricerca di matematica:");

        interfaccia.scrivi("I numeri interi (o numeri interi relativi, o semplicemente, numeri relativi) ");

        interfaccia.scrivi("sono formati dall'unione dei numeri naturali (0,1,2,...) e dei numeri interi negativi (-1,-2,-3,...) ");

        interfaccia.scrivi("costruiti ponendo un segno - davanti ai numeri naturali");

    }

    else if(num==2){

        interfaccia.scrivi("Ricerca di geografia:");

        interfaccia.scrivi("Taranto e' un comune italiano di 202.063 abitanti capoluogo dell'omonima provincia, in Puglia");

        interfaccia.scrivi("Il clima e' dolce e mite.");

        interfaccia.scrivi("Taranto inoltre e' la casa della marina militare italiana.");

    }

    else if(num==3){

        interfaccia.scrivi("Ricerca di storia:");

        interfaccia.scrivi("La prima guerra mondiale fu un conflitto armato");

        interfaccia.scrivi("che coinvolse le principali potenze mondiali e molte di quelle minori. ");

        interfaccia.scrivi("Il conflitto duro' dal 28 luglio 1914 e l'11 novembre 1918.");

    }

}
```

```

    }

else if(num==4){

    interfaccia.scrivi("Ricerca di informatica:");

    interfaccia.scrivi("Un bit e' l'unita di misura dell'informazione(binary
digit,");

    interfaccia.scrivi("definita come la quantita' minima di informazione");

    interfaccia.scrivi("che serve a discernere tra due possibili ");

    interfaccia.scrivi("eventi equiprobabili.");

}

 storia_gioco.insStoria(stringa_comando , "hai deciso di usare il computer");
//Aggiorna storia gioco
}

```

5.4.3.4 *Implementazione azione_88*

```

void Astro::azione_88(){

    int tipo;// creo una variabile in cui mettero il numero della lezione che
sceglierò

    interfaccia.scrivi("Ci sono vari tipi di lezioni che puoi sentire");

    interfaccia.scrivi("1) Sicurezza a lavoro");

    interfaccia.scrivi("2) Medicina");

    interfaccia.scrivi("3) Astronomia ");

    interfaccia.scrivi("4) Meccanica ");

    interfaccia.scrivi("");

    interfaccia.scrivi("Quale scegli?");

    cin >> tipo;// scelgo che lezione seguire
}

```

```
switch(tipo){
```

```
case 1:
```

```
if(tipo==1){// nel caso in cui dovessi scegliere la prima lezione
```

```
    interfaccia.scrivi("LEZIONE SULLA SICUREZZA SUL LAVORO!!");  
    interfaccia.scrivi("");  
    interfaccia.scrivi("La sicurezza sul lavoro e' l'obiettivo di una attivita'  
lavorativa");  
    interfaccia.scrivi("senza l'esposizione per i lavoratori al rischio di  
infortuni/incidenti");  
    interfaccia.scrivi("e senza il rischio di contrarre una malattia  
professionale. ");  
    interfaccia.scrivi("Ecco perche noi raccomandiamo sempre di usare  
TUTA e CASCO per uscire fuori nello spazio.");  
}  
break;
```

case 2:

```
if(tipo==2){// nel caso in cui dovessi scegliere la seconda lezione
```

```
    interfaccia.scrivi("COME USARE IL PRONTO SOCCORSO!!");
```

```
    interfaccia.scrivi("");
```

```
    interfaccia.scrivi("Siamo felici di annunciarvi che la nostra nave  
presenta un pronto soccorso,");
```

```
    interfaccia.scrivi("in cui i nostri operai si possono curare dopo  
aver subito gravi incidenti.");
```

```
    interfaccia.scrivi("Pero' per potersi curare pero' devono essere in  
possesso di una TESSERA SANITARIA.");
```

```
    interfaccia.scrivi("In mancanza di quest'ultima pero' l'operaio  
potra curarsi anche usando dei GETTONI monouso.");
```

```
}
```

break;

case 3:

```

if(tipo==3){// nel caso in cui dovessi scegliere la terza lezione
    interfaccia.scrivi("LEZIONE DI ASTRONOMIA!!");
    interfaccia.scrivi("");
    interfaccia.scrivi("In questo momento ci troviamo vicino ad
una stella che forma la costellazione di Andromeda");
    interfaccia.scrivi("Andromeda e' una costellazione che si
trova nell'emisfero nord vicino a Pegaso.");
    interfaccia.scrivi("La costellazione ha la forma approssimata
di una lettera A");
    interfaccia.scrivi("È famosa soprattutto per la presenza della
galassia di Andromeda nei suoi confini.");
}
break;

```

case 4:

```

if(tipo==4) {// nel caso in cui dovessi scegliere la quarta lezione
    interfaccia.scrivi("LEZIONE DI MECCANICA ");
    interfaccia.scrivi("");
    interfaccia.scrivi("Nel compartimento stagno della nave sulla
pulsantiera ci sono due pulsanti di colori differenti");
    interfaccia.scrivi("Il pulsante Rosso chiude la parete Est e apre la
parete Ovest verso lo spazio esterno;");
    interfaccia.scrivi("Il pulsante Verde chiude la parete Ovest e apre la
parete Est verso il corridoio;");
    interfaccia.scrivi("Attenti a cosa premete");
}
break;

```

default:// nel caso in cui dovessi immettere un valore che non corrisponde a nessuna lezione

```
interfaccia.scrivi("Lezione non esistente");
```

```

    }

tempo=tempo-5;// ogni qual volta che seguirò una lezione verranno sottratte 5 unità
di vita(unità di tempo)

storia_gioco.insStoria(stringa_comando , "hai deciso di seguire una lezione");
//Aggiorna storia gioco

}

```

5.4.3.5 *Implementazione azione_89*

int pr=0;// creo una variabile che mi permetterà di capire se il proiettore è acceso o no

void Astro::azione_89(){

if(pr==0){

```

        interfaccia.scrivi("Hai acceso il proiettore");

        interfaccia.scrivi("MESSAGGIO PROMOZIONALE");

        interfaccia.scrivi("Leggere fa bene alla mente e al corpo");

        interfaccia.scrivi("leggi di piu");

        interfaccia.a_capo();

        pr=1;//proiettore acceso

```

}

else{

 interfaccia.scrivi("Hai già acceso il proiettore"); //MODIFICA ROSA FAGO

}

storia_gioco.insStoria(stringa_comando , "hai acceso il proiettore"); //Aggiorna storia
gioco

}

5.4.3.6 *Implementazione azione_90*

void Astro::azione_90(){

if (pr==0)//se il proiettore è spento fai la seguente azione

{

 interfaccia.scrivi("Il proiettore e' gia' spento");

}

```

else // se il proiettore è acceso stampa messaggio di spegnimento MODIFICA ROSA FAGO
{
    interfaccia.scrivi("hai spento il proiettore");
    pr=0;
}
storia_gioco.insStoria(stringa_comando , "Hai spento il proiettore"); //Aggiorna storia gioco
}

```

5.4.4 Implementazione dell'Aula

In mappa.nav sono state apportate le seguenti modifiche:

- Il numero di luoghi presenti nel gioco è stato modificato da 24 a 25, in modo da poter contenere il nuovo luogo.
- Poiché l'Aula è raggiungibile solo andando a Sud del corridoio sud, è stato modificato il codice di movimento del corridoio sud:
 - *4,All'estremita' sud del corridoio,032500071308,via 1,4,4,via 4,1,1,via 1,7,7*
- È stata aggiunta la riga relativa al nuovo luogo, l'Aula:
 - *25,Nell'Aula,040000000000,via 1,1,1*
- Nella cartella “descrizione” è stato aggiunto un file di testo “25.txt” contenente le descrizioni dell'aula.
- Sono state aggiunte le seguenti righe:
 - *25,4,via 1,nord,4,4,1,1*
 - *4,25,via 1,sud,4,4,1,1*

5.4.5 Correzione Funzionalità usa computer (Andrea Tursi)

Nel luogo aula la funzionalità “usa computer” non veniva riconosciuta durante le fasi di gioco.

```

Provieni dal luogo: 4: All'estremita' sud del corridoio
Sei In nell'aula.
Vedo:
- un proiettore;
- un computer;
- dei libri;
- una lezione in corso.
Tempo residuo: 340
Tempo residuo: 338
Cosa devo fare?
usa computer

- Non capisco.

Sei Di nuovo nell'aula.
Vedo:
- un proiettore;
- un computer;
- dei libri;
- una lezione in corso.
Tempo residuo: 337
Tempo residuo: 335
Cosa devo fare?

```

Per risolvere il problema si è intervenuto sul file Astro.cpp.

Per implementare la funzionalità, era stato inserito nel file il vocabolo “computer” (riga 305) con codice oggetto 78 vocabolario.inserisci("computer", 78);

Inoltre, era stato inserito l’oggetto computer (non trasportabile), facendo riferimento al vocabolo con codice 78 e al luogo Aula con codice 25 (riga 807): oggetti.inserisci(Oggetto("un computer", 78, -25));

A tale oggetto faceva riferimento l’azione “Usa computer”, inserita nel file Astro.cpp alla riga 604

```
azioni.inserisci(2500290078, 87); // "Usa computer"
```

La codifica numerica fa riferimento al luogo 25 (Aula), vocabolo “usa” con codice 29 e vocabolo “computer” con codice 78.

Si è riscontrata la presenza di un altro vocabolo “computer” inserito nel vocabolario a riga 343 vocabolario.inserisci("computer", 99);

Tale vocabolo viene utilizzato per la creazione di diversi oggetti Terminali, sparsi per la mappa.

Allora si è pensato di evitare l’inserimento di un secondo vocabolo “computer”, eliminando la riga di codice apposita, e di modificare il file Astro.cpp in modo tale far riferimento al vocabolo già presente.

Modifica dell’oggetto computer:

```
oggetti.inserisci(Oggetto("un computer", 99, -25));
```

Modifica dell’azione “usa computer” nel luogo Aula:

```
azioni.inserisci(2500290099, 87); // "Usa computer"
```

```

Provieni dal luogo: 4: All'estremita' sud del corridoio
Sei In nell'aula.
Vedo:
- un proiettore;
- un computer;
- dei libri;
- una lezione in corso.
Tempo residuo: 340
Tempo residuo: 338
Cosa devo fare?
usa computer

Ricerca di matematica:
I numeri interi (o numeri interi relativi, o semplicemente, numeri relativi)
sono formati dall'unione dei numeri naturali (0,1,2,...) e dei numeri interi negativi (-1,-2,-3,...)
costruiti ponendo un segno - davanti ai numeri naturali

Sei Di nuovo nell'aula.
Vedo:
- un proiettore;
- un computer;
- dei libri;
- una lezione in corso.
Tempo residuo: 337
Tempo residuo: 335
Cosa devo fare?
usa computer

Ricerca di storia:
La prima guerra mondiale fu un conflitto armato
che coinvolse le principali potenze mondiali e molte di quelle minori.
Il conflitto duro' dal 28 luglio 1914 e l'11 novembre 1918.

```

5.5 UFFICIO POSTALE

5.5.1 Inserimento del luogo "Ufficio Postale"

5.5.1.1 *Ufficiopostale.h*

```
#ifndef UFFICIOPOSTALE_H_
```

```
#define UFFICIOPOSTALE_H_
```

```
#include "luogoufficio.h"
```

```
#include <cstdlib> //funzione random
```

```
using namespace std;
```

```

class UfficioPostale:public luogoufficio {

public:

    UfficioPostale();

    virtual ~UfficioPostale();

    int randomFila(); // genera un numero casuale da 0 a 10 compresi.

    bool get_Suggerimento();

    bool get_Pacco();

    void set_Suggerimento(bool);

    void set_Pacco(bool);

```

```

bool get_ritiratoPacco();

bool get_ritiratoSuggerimento();

void set_ritiratoPacco(bool);

void set_ritiratoSuggerimento(bool);

private:

bool invia_Richiesta_Suggerimento;

bool invia_Richiesta_Pacco;

bool ritirato_Suggerimento;

bool ritirato_Pacco; };

#endif

```

5.5.1.2 *UfficioPostale.cpp*

```
#include "UfficioPostale.h"
```

```

UfficioPostale::UfficioPostale() {

    invia_Richiesta_Pacco=false;
    invia_Richiesta_Suggerimento=false;
    ritirato_Pacco=false;
    ritirato_Suggerimento=false;
}

```

```

UfficioPostale::~UfficioPostale() {
}

```

```

int UfficioPostale::randomFila(){

    return(rand() %11);
}

void UfficioPostale::set_Pacco(bool b){

    invia_Richiesta_Pacco = b ;
}

void UfficioPostale::set_Suggerimento(bool c){

    invia_Richiesta_Suggerimento = c;
}

bool UfficioPostale::get_Pacco(){

```

```

return(invia_Richiesta_Pacco);

}

bool UfficioPostale::get_Suggerimento(){
    return(invia_Richiesta_Suggerimento);

}

void UfficioPostale::set_ritiratoPacco(bool d){
    ritirato_Pacco=d;
}

void UfficioPostale::set_ritiratoSuggerimento(bool e){
    ritirato_Suggerimento=e;
}

bool UfficioPostale::get_ritiratoPacco(){
    return(ritirato_Pacco);
}

bool UfficioPostale::get_ritiratoSuggerimento(){
    return(ritirato_Suggerimento);
}

```

Sarà inoltre necessaria la creazione di una nuova istanza di UfficioPostale in "Gioco.h"

5.5.2 Gioco.h

Riga 44 :

```
#include "UfficioPostale.h"
```

Riga 161 :

```
UfficioPostale ufficiopostale;
```

5.5.3 Codice per inserimento oggetti

Inserimento vocaboli :

5.5.3.1 Astro.cpp

```

vocabolario.inserisci("lettera",37); // da 62
vocabolario.inserisci("pacco", 57); // da 63
vocabolario.inserisci("sportello",99); //da 95
vocabolario.inserisci("documento", 62); //da 61

```

Ogni nuovo oggetto avrà un vocabolo separato per distinguerlo dagli altri oggetti.

Inserimento oggetti :

5.5.3.2 *Astro.cpp*

```
oggetti.inserisci(Oggetto("un documento d'identita",62,6));
oggetti.inserisci(Oggetto("una lettera",37,-99));
oggetti.inserisci(Oggetto("uno sportello telematico",99,-24));
oggetti.inserisci(Oggetto("un pacco",57,-99));
```

Nel vettore `oggetti` occuperanno rispettivamente le posizioni 81,82,83,84.

5.5.4 Nuove azioni

Per l'interazione con i nuovi oggetti sono state inserite le seguenti azioni :

- Guarda Sportello
- Apri pacco
- Leggi lettera

Che saranno rispettivamente le azioni 81,82,83.

5.5.4.1 *Astro.h*

```
void azione_82();//Guarda Ufficio Postale
void azione_83();//Apri Pacco
void azione_84();//Leggi Lettera
```

5.5.4.2 *Astro.cpp*

```
azioni.inserisci(241099,82);//AZIONE 82/InUfficioPostale-guarda-sportello
azioni.inserisci(2257,83);//AZIONE 83/apri-pacco
azioni.inserisci(2537,84);//AZIONE 84/leggi-lettera
```

Azione 82 :

```
void Astro :: azione_82 () {
    std::string risp;

    int p,spedizione,luogo;
    Oggetti inventario;
    p=ufficiopostale.randomFila();

    system("cls");
```

```

interfaccia.scrivi(" ----- ATTENZIONE ----- !");
cout<<" Ci sono "<<p<<" persone in fila, vuoi attendere? [s/n]" ;
cin>>risp;
if(strcmp(risp.c_str(),"s") == 0)

{
    tempo=tempo-p;

int scelta,sceltainvia;
bool uscita,uscitainvia;
uscita=false;
uscitainvia=false;

system("cls");
interfaccia.scrivi("Caricamento del sistema in corso.....");
Sleep(920);
system("cls");
interfaccia.scrivi("\n+-----+");
interfaccia.scrivi("+ Benvenuto nel sistema postale Adventure +");
interfaccia.scrivi("+ Potrai usufruire dei seguenti servizi +");
interfaccia.scrivi("+ Digita: +");
interfaccia.scrivi("+ 1 per accedere ai servizi di invio +");
interfaccia.scrivi("+ 2 per accedere ai servizi di ricezione +");
interfaccia.scrivi("+ 0 per uscire +");
interfaccia.scrivi("+-----+");

while(uscita != true)
{
    cout<<("\n Opzione: ");
    cin>>scelta;
    switch (scelta)
    {
        case 1:
            system("cls");
            interfaccia.scrivi(" Caricamento in corso . . . ");
            Sleep(820);
            system("cls");
            interfaccia.scrivi("\n+-----+");
}

```

```

interfaccia.scrivi("+ Sei nella sezione di invio +");
interfaccia.scrivi("+ Digita: +");
interfaccia.scrivi("+ 1 per richiedere un suggerimento +");
interfaccia.scrivi("+ 2 per richiedere un pacco misterioso +");
interfaccia.scrivi("+ 3 per inviare un pacco in un luogo della nave +");
interfaccia.scrivi("+ 4 per depositare del denaro in banca +");
interfaccia.scrivi("+ 0 per uscire +");
interfaccia.scrivi("+-----+");

while(uscitainvia != true)
{
    cout<<(" \n Opzione: ");
    cin>>sceletainvia;
    switch(sceletainvia)
    {
        case 1:
            if(ufficiopostale.get_Suggerimento() == true)
            {
                system("cls");
                interfaccia.scrivi(" Attendere prego . . . ");
                Sleep(820);
                system("cls");
                interfaccia.scrivi("\n+-----+");
                interfaccia.scrivi("+ Attenzione! Hai gia' richiesto il suggerimento! +");
                interfaccia.scrivi("+ Non puoi utilizzare nuovamente questa funzione +");
                interfaccia.scrivi("+-----+");
            }else
            {
                ufficiopostale.set_Suggerimento(true);
                system("cls");
                interfaccia.scrivi(" Invio in corso . . . ");
                Sleep(820);
                system("cls");
                interfaccia.scrivi("\n+-----+");
                interfaccia.scrivi("+ Richiesta inviata correttamente! +");
            }
    }
}

```

```

interfaccia.scrivi("+ Riceverai a breve la lettera.           +");
interfaccia.scrivi("+ Controlla nella sezione di ricezione.      +");
interfaccia.scrivi("+-----+");

}

uscitainvia=true;
break;

case 2:

if(ufficiopostale.get_Pacco() == true)
{
    system("cls");
    interfaccia.scrivi("Attendere prego . . .");
    Sleep(820);
    system("cls");
    interfaccia.scrivi("\n+-----+");
    interfaccia.scrivi("+ Attenzione! Hai gia' richiesto il pacco!      +");
    interfaccia.scrivi("+ Non puoi utilizzare nuovamente questa funzione      +");
    interfaccia.scrivi("+-----+");
}

else
{
    ufficiopostale.set_Pacco(true);
    system("cls");
    interfaccia.scrivi("Invio in corso . . .");
    Sleep(820);
    system("cls");
    interfaccia.scrivi("\n+-----+");
    interfaccia.scrivi("+ Richiesta inviata correttamente!      +");
    interfaccia.scrivi("+ Riceverai a breve il pacco.           +");
    interfaccia.scrivi("+ Controlla nella sezione di ricezione.      +");
    interfaccia.scrivi("+-----+");
}

uscitainvia=true;
break;

```

case 3:

```
    system("cls");
    interfaccia.scrivi("Caricamento in corso . . .");
    Sleep(820);
    system("cls");
    interfaccia.scrivi("\n+-----+");
    interfaccia.scrivi("+ Scegli dall'inventario l'oggetto da inviare.      +");
    interfaccia.scrivi("+ Inserendo il relativo codice spedizione.      +");
    for (int i=1; i<= oggetti.get_n_oggetti();i++)
    {
        if(oggetti.get_oggetto(i).get_luogo() == 0)
        {
            cout<<"\n - "<<oggetti.get_oggetto(i).get_nome()<<". codice spedizione: --> "<<i;
        }
    }
    cout<<"\n Codice spedizione: ";
    cin>>spedizione;
    interfaccia.scrivi("+ Inserisci il codice del luogo di destinazione.      +");
    cout<<"\n - Sala controllo del reattore codice luogo      --> 8 +";
    cout<<"\n - Nella cabina del secondo pilota      --> 5 +";
    cout<<"\n Codice luogo: ";
    cin>>luogo;
    interfaccia.scrivi("+-----+");
    if (luogo == 8 || luogo == 5)
    {
        oggetti.set_luogo(spedizione,luogo);
        system("cls");
        interfaccia.scrivi("Invio in corso . . .");
        Sleep(820);
        system("cls");
        interfaccia.scrivi("\n+-----+");
        interfaccia.scrivi("+ Operazione completata!      +");
        interfaccia.scrivi("+-----+");
    }
}
```

else

```

{
    system("cls");
    interfaccia.scrivi("Invio in corso . . .");
    Sleep(820);
    system("cls");
    interfaccia.scrivi("\n+-----+");
    interfaccia.scrivi("+ Operazione annullata! +");
    interfaccia.scrivi("+ Codice luogo non valido +");
    interfaccia.scrivi("+-----+");
}

uscitainvia=true;
break;

case 4:
    system("cls");
    interfaccia.scrivi("Caricamento in corso . . .");
    Sleep(820);
    system("cls");
    azione_56();
    uscitainvia=true;
    break;

case 0:
    uscitainvia=true;
    break;

default:
    tempo=tempo-1;
    interfaccia.scrivi("Non ho capito");
}

}

uscita=true;
break;

case 2:

```

```

if(oggetti.get Oggetto(81).get Luogo() == 0) //get Oggetto(x,y) x da 80 a 81 Galeandro
{
    system("cls");
    interfaccia.scrivi(" Caricamento in corso . . .");
    Sleep(820);
    system("cls");
    interfaccia.scrivi("\n+-----+");
    interfaccia.scrivi("+ Sei nella sezione di ricezione. +");
    if(ufficiopostale.get_Pacco()== false && ufficiopostale.get_Suggerimento()==false)
    {
        interfaccia.scrivi("+ Nessun oggetto da ritirare. +");
        interfaccia.scrivi("\n+-----+");
    }

    if(ufficiopostale.get_Pacco() == true && ufficiopostale.get_ritiratoPacco() == false)
    {
        interfaccia.scrivi("+ È arrivato un pacco per te. +");
        interfaccia.scrivi("+ Controlla il tuo inventario. +");
        interfaccia.scrivi("+-----+");
        ufficiopostale.set_ritiratoPacco(true);
        oggetti.set_Luogo(84,0); //set_Luogo(x,y) x da 83 a 84 Galeandro
    }
    else if(ufficiopostale.get_ritiratoPacco() == true)
    {
        interfaccia.scrivi("+ Hai già ritirato il pacco +");
    }

    if(ufficiopostale.get_Suggerimento()== true && ufficiopostale.get_ritiratoSuggerimento()==false)
    {
        interfaccia.scrivi("+È arrivata una lettera per te. +");
        interfaccia.scrivi("+ Controlla il tuo inventario. +");
        interfaccia.scrivi("+-----+");
        ufficiopostale.set_ritiratoSuggerimento(true);
        oggetti.set_Luogo(82,0); //set_Luogo(x,y) x da 81 a 82 Galeandro
    }
}

```

```

else if(ufficiopostale.get ritiratoSuggerimento()== true)
{
    interfaccia.scrivi(" + Hai già ritirato la lettera + ");
}
}

else
{
    system("cls");
    interfaccia.scrivi("\n+-----+");
    interfaccia.scrivi(" + Attenzione! + ");
    interfaccia.scrivi(" + Recupera prima il tuo documento d'identità + ");
    interfaccia.scrivi(" + e avrai accesso alla sezione desiderata. + ");
    interfaccia.scrivi(" +-----+");

}
uscita=true;
break;
```

case 0:

```

system("cls");
interfaccia.scrivi("\n+-----+");
interfaccia.scrivi(" + Grazie e Arrivederci! + ");
interfaccia.scrivi(" +-----+");

uscita=true;
break;
```

default:

```

tempo=tempo-1;
cout<<"\n non capisco";
```

}

Azione 83 :

```

void Astro :: azione_83 () {
    if(oggetti.get_oggetto(84).get_luogo() == 0)//get_oggetto(x) x da 83 a 84 Galeandro
    {
        interfaccia.scrivi("Hai aperto il pacco misterioso.");
        interfaccia.scrivi("Messaggio:");
        interfaccia.scrivi("Ecco a te il doppione della chiave dell'armadietto del secondo pilota!..");
        interfaccia.scrivi("Cerca di non perderla!");
        interfaccia.scrivi("Controlla il tuo inventario.");
        oggetti.set_luogo(24,0);//set_luogo(x,y) x da 23 a 24 Galeandro
        oggetti.set_luogo(84,-99);//set_luogo(x,y) x da 83 a 84 Galeandro
    }
    else
    {
        interfaccia.scrivi("Non capisco.");
    }
}

```

Azione 84 :

```

void Astro :: azione_84 () {
    if(oggetti.get_oggetto(82).get_luogo() == 0)//get_oggetto(x) x da 81 a 82 Galeandro
    {
        interfaccia.scrivi("Hai aperto la lettera.");
        interfaccia.scrivi("Messaggio:");
        interfaccia.scrivi("La chiave dell'armadietto del secondo pilota potrebbe essere andata perduta!..");
        interfaccia.scrivi("Per fortuna che alla base hanno sempre i pezzi di ricambio!");
    }
    else
    {
        interfaccia.scrivi("Non capisco.");
    }
}

```

5.5.5 Modifiche al progetto di Bellanova

Oltre alle aggiunte fatte al progetto di D'Andria\Dresda sono state necessarie alcune modifiche al progetto di Bellanova :

- Documento d'identità : Vocabolo\oggetto da 61 a 62
- Lettera : Vocabolo\oggetto da 62 a 37
- Pacco : Vocabolo\oggetto da 63 a 57
- Sportello telematico : Vocabolo\oggetto da 95 a 99
- Ufficio postale : Luogo da 23 a 24
- Azioni : da 80,81,82 a 81,82,83 con relative modifiche dovute dalle posizioni degli oggetti
- Posizione degli oggetti : da 80,81,82,83 a 81,82,83,84

5.6 DIALOGHI

5.6.1 Implementazione delle variabili

5.6.1.1 *Personaggi.h*

Includiamo la classe Dialogo

```
7 | #include "Dialogo.h" //Aggiunta D'Andria Dresden al progetto di Federica Forte
```

Aggiungiamo l'attributo dialogo ai personaggi

```
51 | Dialogo dialogo; //Aggiunta D'Andria Dresden al progetto di Federica Forte
```

5.6.2 Implementazione dei metodi

5.6.2.1 *Personaggi.h*

Aggiungiamo il metodo per settare un dialogo ad un personaggio

```
24 | void setDialogo(Dialogo*); //aggiunta D'Andria Dresden al progetto di Federica Forte
```

Aggiungiamo il metodo per prendere il dialogo di un personaggio

```
31 | Dialogo* getDialogo(); //aggiunta D'Andria Dresden al progetto di Federica Forte
```

5.6.2.2 *Personaggi.cpp*

5.6.2.2.1 Metodo setDialogo:

```
145 | void Personaggi::setDialogo(Dialogo *dialogodainserire){
146 |     this->dialogo=*dialogodainserire;
147 | }
```

5.6.2.2.2 Metodo getDialogo:

```
151 | Dialogo* Personaggi::getDialogo(){
152 |     return (&this->dialogo);
153 | }
```

5.6.2.3 *Gioco.h*

```
260 | Dialogo caricaDialoghiDaFile(string);
261 | void convertiRiga(Dialogo*,string,string);
262 | int caricastatodialogo(string);
263 | int leggistato(int*,string, string);
264 | void scrivistatosufile(string, int);
265 | bool verificanome(string,string);
```

```

266     void salvaStatiDialoghi();
267     void caricaStatiDialoghi();

```

5.6.2.4 Gioco.cpp

5.6.2.4.1 Metodo caricaDialoghiDaFile:

Questo metodo prende in ingresso il nome del personaggio e cerca all'interno del file Dialoghi.txt il dialogo associato a quel personaggio; legge il file riga per riga chiamando la procedura convertiRiga.

```

2483     Dialogo Gioco::caricaDialoghiDaFile(string nome){
2484         Dialogo dialogo= Dialogo(nome);
2485         string rigafile;
2486         ifstream filedialoghi("Dialoghi.txt", ios::in);
2487         while (filedialoghi.get()!='?'){
2488             getline(filedialoghi,rigafile);
2489             convertiRiga(&dialogo,nome,rigafile);
2490         }
2491         filedialoghi.close();
2492         return dialogo;
2493     }

```

5.6.2.4.2 Metodo convertiRiga:

Abbiamo strutturato il file Dialoghi.txt in modo tale che ogni riga si presenti come quella in esempio:

*-Cid|1*Ciao, sono Cid!!\$\$Credi in Dio?\$1:si\$2:no\$0:interrompi dialogo\$#*

5.6.2.4.2.1 Significato della riga

- : Simbolo iniziale della riga;
- Cid** : Nome del personaggio che deve iniziare con la lettera maiuscola;
- | : Separatore tra il nome del personaggio e il codice domanda;
- 1** : Codice della domanda;
- * : Separatore tra il codice domanda e la frase;
- \$: Simbolo che verrà convertito in \n;
- # : Simbolo usato al termine della riga.

Il metodo riceve in ingresso il puntatore a dialogo, il nome da cercare e la riga del file da analizzare e convertire.

Dopo aver analizzato la riga in base alla sintassi prima descritta, se il nome del personaggio contenuto nella riga del file coincide con il nome che stiamo cercando, inserisce nel dialogo di quel personaggio la domanda composta dal codice e dalla frase.

```

2503     void Gioco::convertiRiga(Dialogo *dialogo,string nome,string rigafile){
2504         string nomepersonaggio="";
2505         char statodialogostr[20];
2506         for(int j=0; j<20; j++)
2507             statodialogostr[j]='\0';
2508         int statodialogo;
2509         string frasedialogo="";
2510         int indice=0;
2511         while(rigafile[indice]!='|'){

```

```

2512     nomepersonaggio=nomepersonaggio+rigafile[indice]; //leggo il nome del personagg
2513     io
2514         indice++;
2515     }
2516     indice++;
2517     int i=0;
2518     while(rigafile[indice]!='*'){
2519         statodialogostr[i]=rigafile[indice]; // leggo lo stato del dialogo
2520         indice++;
2521         i++;
2522     }
2523     statodialogo=atoi(statodialogostr);
2524     indice++;
2525     while(rigafile[indice]!='#'){
2526         if(rigafile[indice]=='$')
2527             frasedialogo=frasedialogo+'\n';
2528         else
2529             frasedialogo=frasedialogo+rigafile[indice]; // leggo la frase da inserire n
el nodo Domanda
2530         indice++;
2531     }
2532     if (nome==nomepersonaggio)
2533         dialogo-
>inseriscidomanda(statodialogo,frasedialogo); //inserisce la Domanda nel Dialogo
2534 }
```

5.6.2.5 Metodo caricastatodialogo:

Metodo chiamato in ' Gioco::az_parla ' quando il giocatore inserisce il comando ' parla con NomePersonaggio ' se al personaggio è stato assegnato un dialogo.

Riceve in ingresso il nome del personaggio e restituisce lo stato del dialogo letto dal file ' Statidialoghi.txt '.

Il metodo legge il file riga per riga dando la riga letta come parametro del metodo ' leggistato '.

```

2700 int Gioco::caricastatodialogo(string nome){
2701     int stato=1;
2702     string rigafile;
2703     ifstream filestati("Statidialoghi.txt", ios::in);
2704     if(!filestati.is_open())
2705         cout << "File Statidialoghi.txt non trovato";
2706     else{
2707         while (filestati.get()!='?'){
2708             getline(filestati,rigafile);
2709             leggistato(&stato,nome,rigafile); //legge lo stato all'interno della riga d
el file
2710         }
2711     }
2712     filestati.close();
2713     return stato;
2714 }
```

5.6.2.6 Metodo leggistato:

Metodo che riceve in ingresso il puntatore a ' stato ', il nome del personaggio cercato e la riga del file letta restituendo un numero intero corrispondente allo stato del dialogo salvato nel file ' Statidialoghi.txt '.

```

2722 int Gioco::leggistato(int *stato,string nome, string rigafile){
2723     string nomepersonaggio;
2724     char statodialogostr[20];
2725     for(int j=0; j<20; j++)
2726         statodialogostr[j]=NULL;
2727     int statodialogo;
2728     int indice=0;
2729     while(rigafile[indice]!='|'){

182
```

```

2730         nomepersonaggio=nomepersonaggio+rigafile[indice];
2731         indice++;
2732     }
2733     indice++;
2734     int i=0;
2735     while(rigafile[indice]!='#'){
2736         statodialogostr[i]=rigafile[indice];
2737         indice++;
2738         i++;
2739     }
2740     statodialogo=atoi(statodialogostr);
2741     if (nome==nomepersonaggio)
2742         *stato=statodialogo;
2743 }
```

5.6.2.7 *Metodo scrivistatosofile:*

Questo metodo riceve in ingresso il nome del personaggio e lo stato del dialogo nel momento in cui il dialogo è stato interrotto.

Crea un nuovo file di testo chiamato ' Nuovofile.txt ' aprendolo in modalità scrittura.

Legge riga per riga il file ' Statidialoghi.txt ', controlla il nome del personaggio all'interno della riga letta:

- se il nome coincide con il nome cercato
 - crea una stringa che contiene -NomePersonaggio|StatoDialogo#
 - scrive la stringa sul file ' Nuovofile.txt '
- se il nome non coincide con il nome cercato
 - copia semplicemente la riga letta dal file ' Statidialoghi.txt ' nel file ' Nuovofile.txt '

Infine elimina il file ' Statidialoghi.txt ' e rinomina con quel nome il file ' Nuovofile.txt '

```

2752 void Gioco::scrivistatosofile(string nome, int stato){
2753     ifstream file("Statidialoghi.txt");
2754     ofstream nuovofile("Nuovofile.txt");
2755     string rigafile;
2756     string nuovariga;
2757     bool riganome=false;
2758     if(!file)
2759         cout << "File Statidialoghi.txt non trovato";
2760     else{
2761         while(file.get()!='?'){
2762             getline(file,rigafile);
2763             riganome=verificanome(nome,rigafile);
2764             if(riガname){
2765                 stringstream statostream;
2766                 statostream<<stato;
2767                 string statostring=statostream.str();
2768                 nuovariga='-' + nome + '|' + statostring + '#';
2769                 nuovofile << nuovariga << '\n';
2770             }
2771             else
2772                 nuovofile << '-' << rigafile << '\n';
2773         }
2774     }
2775     nuovofile << '?';
2776     file.close();
2777     nuovofile.close();
2778     std::remove("Statidialoghi.txt");
2779     int result= rename("Nuovofile.txt", "Statidialoghi.txt");
2780 }
```

5.6.2.8 *Metodo verificanome:*

Il metodo prende in ingresso il nome del personaggio cercato e la riga letta dal file 'Statidialoghi.txt' restituendo un booleano che avrà il valore del confronto dei due nomi.

```
2793     bool Gioco::verificanome(string nome, string rigafile){  
2794         string nomepersonaggio;  
2795         int indice=0;  
2796         while(rigafile[indice]!='|'){  
2797             nomepersonaggio=nomepersonaggio+rigafile[indice];  
2798             indice++;  
2799         }  
2800         return(nome==nomepersonaggio);  
2801     }
```

5.6.2.9 *Metodo salvaStatiDialoghi:*

Il metodo copia il contenuto del file 'Statidialoghi.txt' in un file chiamato 'AstroStatiDialoghi.txt' nel momento in cui il giocatore digita il comando ' save ' per salvare la partita in corso.

```
1338     void Gioco::salvaStatiDialoghi(){  
1339         ofstream nuovofile("Astrostatidialoghi.txt");  
1340         ifstream file("Statidialoghi.txt");  
1341         string rigafile;  
1342         while(file.get()!='?'){  
1343             getline(file,rigafile);  
1344             nuovofile << '-' << rigafile << '\n';  
1345         }  
1346         nuovofile << '?';  
1347         file.close();  
1348         nuovofile.close();  
1349     }
```

5.6.2.10 *Metodo caricaStatiDialoghi:*

Il metodo copia il contenuto del file 'AstroStatiDialoghi.txt' nel file chiamato 'StatiDialoghi.txt' nel momento in cui il giocatore digita il comando ' load ' per caricare un salvataggio avvenuto precedentemente.

```
1419     void Gioco::caricaStatiDialoghi(){  
1420         ifstream file("Astrostatidialoghi.txt");  
1421         ofstream nuovofile("Statidialoghi.txt");  
1422         string rigafile;  
1423         while(file.get()!='?'){  
1424             getline(file,rigafile);  
1425             nuovofile << '-' << rigafile << '\n';  
1426         }  
1427         nuovofile << '?';  
1428         file.close();  
1429         nuovofile.close();  
1430     }
```

5.6.3 Modifiche dei metodi già presenti in Gioco.cpp

Le parti di codice evidenziate all'interno del metodo, sono le istruzioni aggiunte per implementare i metodi creati.

5.6.3.1 *Save*

```
1358     void Gioco::load(){  
1359         int i;  
1360         int valore;  
1361         int slot; //CICALA GIACOMO  
1362         svuotaInsieme(slotmachine); //CICALA GIACOMO
```

```

1363     ifstream file(fStringa.c_str(), ios::in);
1364     //modifiche effettuate da D'Andria Dresda sul controllo del caricamento da file
1365     ifstream astrostatidialoghi("Astrostatidialoghi.txt");
1366     bool astrostatidialoghiaperto=astrostatidialoghi.good();
1367     bool fileaperto=file.good();
1368     if(fileaperto || astrostatidialoghiaperto){
1369         interfaccia.scrivi("Ripristino partita..."); 
1370         if(fileaperto){
1371             for (i = 1; i <= oggetti.get_n_oggetti(); i++){
1372                 file >> valore;
1373                 oggetti.set_luogo(i, valore);
1374             }
1375             file >> luogo_attuale;
1376             file >> tempo;
1377             file >> passo_soluzione;
1378             load_specifiche(file);
1379             bacheca.CaricaBacheca(file);
1380             file.close();
1381         }
1382         if(astrostatidialoghiaperto)
1383             caricaStatiDialoghi(); //modifiche D'Andria Dresda
1384         ifstream jukeLuci("JukeLuci.txt");
1385         if (jukeLuci.good() )
1386             caricaJukeBoxLuci();
1387     }
1388     else
1389         interfaccia.scrivi("Non ci sono partite salvate!");
1390     //INIZIO MODIFICHE CICALA GIACOMO
1391     file >> numEuro;
1392     for(int j=1; j<=numEuro; j++){
1393         file >> slot;
1394         slotmachine.inserisci(slot);
1395     }
1396     //fine modifiche CICALA GIACOMO
1397 }
1398 }
```

5.6.3.2 switch_enigmi

```

1511 // modifica zagaria -- aggiunto il paramentro per la gestione dei personaggi
1512 void Gioco::switch_enigmi(int a, Mappa &M, Pila<stato_comando>* p, Personaggi* pg){
1513     bool parlato; //modifica D'Andria Dresda al progetto di Federica Forte
1514     if(leggienigma()){
1515         cout<<"\nComplimenti hai indovinato l'enigma puoi proseguire" << endl;
1516         cout<<"\n";
1517         //Modifica Pmf: fin qui.
1518         //INIZIO MODIFICA MICHELE ALBANO
1519         if(!sFisico.feritaEvitata(rand()))//Modifica DAVIDE MANTELLINI
1520             Aggiorna_Ferite(); //Generatore Ferite
1521         if(Salute.GetStatoSalute()==0) //Se lo Stato della Salute è a 0, fine dell'avve
1522             ntura
1523             morto();
1524             interfaccia.a_capo();
1525             //FINE MODIFICA MICHELE ALBANO
1526             switch (a){
1527                 case 1:
1528                     parlato=false; //aggiunta D'Andria Dresda dal progetto di Gallo in quello d
1529                     i Federica Forte
1530                     direzioni(M);
1531                     break;
1532                 case 2:
1533                     prendi();
1534                     break;
1535                 case 3:
1536                     lascia();
1537                     break;
1538                 case 4:
```

```

1537         guarda();
1538         break;
1539     case 5:
1540         save();
1541         break;
1542     case 6:
1543         load();
1544         break;
1545     case 7:
1546         cosa();
1547         break;
1548     case 8:
1549         navigatore(M);
1550         system("cls");
1551         //system("clear"); //linux os //Modificato da ANTONIO PASTORELLI
1552         break;
1553     case 9:
1554         indietro(p);
1555         break;
1556     case 39:
1557         macchinadeltempo();
1558         break;
1559         //aggiunta D'Andria Dresden dal progetto di Federica Forte
1560     case 99:
1561         if(parlato==true)
1562             cout<<"Ci hai gia' parlato"<<endl;
1563         else
1564         {
1565             //modifica D'Andria Dresden
1566             az_parla(*pg);
1567             if(pg->getNome()!="Cid" && pg->getNome()!="Vincent" && pg-
1568             >getNome()!="Kail")
1569                 parlato=true;
1570         }
1571         break;
1572         //fine aggiunta D'Andria Dresden
1573     case 40:
1574         visualizza_bacheca();
1575         system("cls");
1576         //system("clear"); //linux os //Modificato da ANTONIO PASTORELLI
1577         break;
1578     case 41:
1579         benzina();
1580         break;
1581     case 42:
1582         usa_motorino();
1583         break;
1584         // inizio modifiche zagaria
1585     case 43:
1586         stringa_risposta = "Hai parlato con il personaggio presente nella stanza.";
1587         //aggiungo alla storia
1588         storia_gioco.insStoria(stringa_comando, stringa_risposta);
1589         az_parla(*pg);
1590         break;                                //fine modifiche
1591         //Modifica PMF(agenda)
1592     case 46:
1593         azione_46();
1594         break;
1595     case 47:
1596         azione_47();
1597         break;
1598     case 48:
1599         azione_48();
1600         break;
1601         //Modifica PMF: fin qui.
1602         //START DAMONE
1603         //Modifica Francesco De Giorgio

```

```

1602         case 49:
1603             consulta(); //inserita la consultazione della guida
1604             //Modifica De Giorgio : fin qui.
1605             break;
1606             //END DAMONE
1607             default: // AZIONI SPECIFICHE
1608                 if (!esegui_specifiche(a,M))
1609                     interfaccia.scrivi("AZIONE " + a); // test
1610             }
1611         }
1612         //Modifica PMF(enigmi)
1613     else{
1614         cout << "\nHai sbagliato la risposta stai perdendo tempo," << endl;
1615         cout << "ridigita il comando per avere un altro enigma" << endl;
1616         tempo--;
1617     }
1618 }
```

5.6.3.3 esegui

```

1637 // MODIFICA zagaria -
1638 - modifica della funzione esegui con l' aggiunta del parametro personaggi
1639 void Gioco::esegui(int a, Mappa &M, Pila<stato_comando>* p, bool si, Personaggi* pg){
1640
1641     if(si)
1642         switch_enigmi(a,M,p,pg); // modifica zagaria -- aggiunto il parametro pg
1643     else{
1644         bool parlato; //Aggiunta D'Andria Dresda
1645         //INIZIO MODIFICA MICHELE ALBANO
1646         Aggiorna_Ferite(); //Generatore Ferite
1647         if(Salute.GetStatoSalute()==0) //Se lo Stato della Salute è a 0, fine dell'avve
1648             morto();
1649         interfaccia.a_capo();
1650         //FINE MODIFICA MICHELE ALBANO
1651         switch (a){
1652             case 1:
1653                 parlato = false; //Aggiunta D'Andria Dresda
1654                 direzioni(M);
1655                 break;
1656             case 2:
1657                 prendi();
1658                 break;
1659             case 3:
1660                 lascia();
1661                 break;
1662             case 4:
1663                 guarda();
1664                 break;
1665             case 5:
1666                 save();
1667                 break;
1668             case 6:
1669                 load();
1670                 break;
1671             case 7:
1672                 cosa();
1673                 break;
1674             case 8:
1675                 navigatore(M);
1676                 system("cls");
1677                 //system("clear"); //linux os //Modificato da ANTONIO PASTORELLI
1678                 break;
1679             case 9:
1680                 indietro(p);
1681                 break;
1682             case 39:
```

```

1682         macchinadeltempo();
1683         break;
1684     //aggiunta D'Andria Dresda
1685     case 99:
1686         if(parlato==true)
1687             cout<<"Ci hai gia' parlato"<<endl;
1688         else{
1689             az_parla(*pg);
1690             if(pg->getNome()!="Cid" && pg->getNome()!="Vincent" && pg-
1691 >getNome()!="Kail") //modifica D'Andria Dresda
1692                 parlato=true;
1693         }
1694         break;
1695     //fine aggiunta D'Andria Dresda
1696     case 40:
1697         visualizza_bacheca();
1698         system("cls");
1699         //system("clear"); //linux os //Modificato da ANTONIO PASTORELLI
1700         break;
1701     case 41:
1702         benzina();
1703         break;
1704     case 42:
1705         usa_motorino();
1706         break;
1707         // inizio modifiche zagaria -
1708         - definisce la comunicazione con il personaggio nella storia
1709     case 43:
1710         stringa_risposta = "Hai parlato con il personaggio presente nella stanza.";
1711     //aggiungo alla storia
1712         storia_gioco.insStoria(stringa_comando, stringa_risposta);
1713         az_parla(*pg);
1714         break;
1715     //fine modifiche
1716     //Modifica PMF(agenda)
1717     case 46:
1718         azione_46();
1719         break;
1720     case 47:
1721         azione_47();
1722         break;
1723     case 48:
1724         azione_48();
1725         break;
1726         //Modifica PMF: fin qui.
1727         //START DAMONE
1728         //Modifica Francesco De Giorgio
1729     case 49:
1730         consulta(); //inserita la consultazione della guida
1731         //Modifica De Giorgio : fin qui.
1732         break;
1733     //END DAMONE
1734     default: // AZIONI SPECIFICHE
1735         if (!esegui_specifiche(a,M))
1736             interfaccia.scrivi("AZIONE " + a); // test
1737     }
1738 }
1739 }
```

5.6.3.4 initPers

```

2208     void Gioco::initPers(Dizionario<int, Personaggi> *insPersonaggi)
2209     {
2210
2211         Personaggi p1,p2,p3,p4,p5,p6,p7,p8,p9,p10,p11,p12,p13,p14,p15,p16,p17,p18,p19,p20,p
2212         21,p22,p23/*,p24,p25,p26,p27,p28,p29,p30*/,
```

```

2212          p31,p32,p33,/*p34,p35,p36,p37,p38,p39,p40,*/p41,p42,p43,/*p44,p45,p46,p4
2213          7,p48,p49,p50,*/*p51,p52,p53,/*p54,p55,p56,p57,p58,p59,p60,*/
2214          p61,p62,p63,/*p64,p65,p66,p67,p68,p69,p70,*/p71,p72,p73,/*p74,p75,p76,p7
2215          7,p78,p79,p80,*/*p81,p82,p83,/*p84,p85,p86,p87,p88,p89,p90,*/
2216          p91,p92,p93/*,p94,p95,p96,p97,p98,p99,p100*/;
2217
2218 //AGGIUNTA DEI PERSONAGGI PER I DIALOGHI BASILE ANTONIO
2219 Personaggi prof_Mara, prof_Clara, alunno_Davide;
2220
2221 //Luogo 1 1-10
2222
2223     p1.setNome("Kain");
2224     p1.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2225     ");
2226     p1.setLuogo(1);
2227     insPersonaggi->inserisci(1, p1);
2228     p2.setNome("Tom");
2229     p2.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2230     ");
2231     p2.setOgg(true,51);
2232     p2.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2233     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2234     ile");
2235     p2.setLuogo(1);
2236     insPersonaggi->inserisci(2, p2);
2237     p3.setNome("Jim");
2238     p3.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2239     ");
2240     p3.setLuogo(1);
2241     insPersonaggi->inserisci(3, p3);
2242     p4.setNome("Adam");
2243     p4.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2244     ");
2245     p4.setOgg(true,51);
2246     p4.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2247     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2248     ile");
2249     p4.setLuogo(1);
2250     insPersonaggi->inserisci(4, p4);
2251     p5.setNome("Paul");
2252     p5.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2253     ");
2254     p5.setLuogo(1);
2255     insPersonaggi->inserisci(5, p5);
2256     p6.setNome("Raoul");
2257     p6.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2258     ");
2259     p6.setOgg(true,51);
2260     p6.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2261     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2262     ile");
2263     p6.setLuogo(1);
2264     insPersonaggi->inserisci(6, p6);
2265     p7.setNome("Max");
2266     p7.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2267     ");
2268     p7.setLuogo(1);
2269     insPersonaggi->inserisci(7, p7);
2270     p8.setNome("Sam");
2271     p8.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2272     ");
2273     p8.setOgg(true,51);
2274     p8.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2275     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2276     ile");
2277     p8.setLuogo(1);
2278     insPersonaggi->inserisci(8, p8);

```

```

2261     p9.setNome("Yan");
2262     p9.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala lì");
2263     p9.setLuogo(1);
2264     insPersonaggi->inserisci(9, p9);
2265     p10.setNome("Arthur");
2266     p10.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala lì");
2267     p10.setOgg(true,51);
2268     p10.setFrasiconOgg("Visto che hai la tuta puoi andare, qui non c'e altro di importante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di utile");
2269     p10.setLuogo(1);
2270     insPersonaggi->inserisci(10, p10);
2271
2272     //Luogo 2 11-20
2273     p11.setNome("Cloud");
2274     p11.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'");
2275     p11.setLuogo(2);
2276     insPersonaggi->inserisci(11, p11);
2277     p12.setNome("Cole");
2278     p12.setFrasi("Controlla la 'Cabina del secondo pilota', potresti trovare qualcosa di utile","La 'Cabina del secondo pilota' e vuota, e inutile entrarci");
2279     p12.setLuogo(2);
2280     insPersonaggi->inserisci(12, p12);
2281     p13.setNome("Zack");
2282     p13.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'");
2283     p13.setOgg(true,55);
2284     p13.setFrasiconOgg("Hai il 'manuale',il reattore deve essere spento, svelto","Il 'manuale' non ti servira' a nulla, lascialo e cerca una soluzione al problema");
2285     p13.setLuogo(2);
2286     insPersonaggi->inserisci(13, p13);
2287     p14.setNome("Gretro");
2288     p14.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'");
2289     p14.setLuogo(2);
2290     insPersonaggi->inserisci(14, p14);
2291     p15.setNome("Leroy");
2292     p15.setFrasi("Controlla la 'Cabina del secondo pilota', potresti trovare qualcosa di utile","La 'Cabina del secondo pilota' e vuota, e inutile entrarci");
2293     p15.setLuogo(2);
2294     insPersonaggi->inserisci(15, p15);
2295     p16.setNome("Tony");
2296     p16.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'");
2297     p16.setOgg(true,55);
2298     p16.setFrasiconOgg("Hai il 'manuale',il reattore deve essere spento, svelto","Il 'manuale' non ti servira' a nulla, lascialo e cerca una soluzione al problema");
2299     p16.setLuogo(2);
2300     insPersonaggi->inserisci(16, p16);
2301     p17.setNome("Timmy");
2302     p17.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'");
2303     p17.setLuogo(2);
2304     insPersonaggi->inserisci(17, p17);
2305     p18.setNome("Eric");
2306     p18.setFrasi("Controlla la 'Cabina del secondo pilota', potresti trovare qualcosa di utile","La 'Cabina del secondo pilota' e vuota, e inutile entrarci");
2307     p18.setLuogo(2);
2308     insPersonaggi->inserisci(18, p18);
2309     p19.setNome("Dan");
2310     p19.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'");
2311     p19.setOgg(true,55);

```

```

2312     p19.setFrasiconOgg1("Hai il 'manuale',il reattore deve essere spento, svelto","Il 'manuale' non ti servira' a nulla, lascialo e cerca una soluzione al problema");
2313     p19.setLuogo(2);
2314     insPersonaggi->inserisci(19, p19);
2315     p20.setNome("Chuck");
2316     p20.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'", "La 'tuta' si trova nella 'sala di controllo del reattore'");
2317     p20.setLuogo(2);
2318     insPersonaggi->inserisci(20, p20);
2319
2320     //luogo 3 21-30
2321     p21.setNome("Vincent");
2322     p21.setFrasi("A nord trovi la 'Cabina di pilotaggio'", "A sud trovi la 'Cabina di pilotaggio'");
2323     p21.setLuogo(3);
2324     //modifiche D'Andria Dresden aggiunta dialogo a Vincent
2325     Dialogo dialogovincent=Dialogo("Vincent");
2326     dialogovincent=caricaDialoghiDaFile("Vincent");
2327     p21.setDialogo(&dialogovincent);
2328
2329     //fine modifiche
2330     insPersonaggi->inserisci(21, p21);
2331     p22.setNome("Kail");
2332     p22.setFrasi("A sud trovi 'il compartimento stagno' e la 'sala del reattore'", "A sud trovi la 'Cabina di pilotaggio'");
2333     p22.setLuogo(3);
2334     //modifiche D'Andria Dresden aggiunta dialogo a Vincent
2335     Dialogo dialogokail=Dialogo("Kail");
2336     dialogokail=caricaDialoghiDaFile("Kail");
2337     p22.setDialogo(&dialogokail);
2338     //fine modifiche
2339
2340     insPersonaggi->inserisci(22, p22);
2341
2342     p23.setNome("Joe");
2343     p23.setFrasi("Cerca il 'casco'", "Il 'casco' e inutile");
2344     p23.setOgg(true,50,51);
2345     p23.setFrasiconOgg1("Cerca la 'tuta'", "la 'tuta' non ti serve");
2346     p23.setFrasiconOgg2("Cerca il 'secondo pilota'", "Il 'secondo pilota' e nella sua cabina");
2347     p23.setLuogo(3);
2348     insPersonaggi->inserisci(23, p23);
2349
2350     //luogo 4 31-40
2351     p31.setNome("Garnet");
2352     p31.setFrasi("Ho visto il 'secondo pilota' andare verso il 'compartimento stagno' a est", "Il secondo pilota non e' passato di qui");
2353     p31.setLuogo(4);
2354     insPersonaggi->inserisci(31, p31);
2355     p32.setNome("Shaun");
2356     p32.setFrasi("Ho visto il 'secondo pilota' andare verso il 'compartimento stagno' a est", "Il secondo pilota non e' passato di qui");
2357     p32.setOgg(true,50,51);
2358     p32.setFrasiconOgg1("Cerca la 'tuta' prima di uscire dall'astronave", "Corri nel 'compartimento stagno' e cerca il 'secondo pilota'");
2359     p32.setFrasiconOgg2("Corri nel 'compartimento stagno' e cerca il 'secondo pilota'", "Lascia la 'tuta' prima di andare");
2360     p32.setLuogo(4);
2361     insPersonaggi->inserisci(32, p32);
2362     p33.setNome("Frank");
2363     p33.setFrasi("Ho visto il 'secondo pilota' andare verso il 'compartimento stagno' a est", "Il secondo pilota non e' passato di qui");
2364     p33.setOgg(true,54);
2365     p33.setFrasiconOgg1("Vai nella 'cabina del secondo pilota'", "La 'Chiave' che hai preso non va bene, cercane un'altra");
2366     p33.setLuogo(5);
2367     insPersonaggi->inserisci(33, p33);

```

```

2368
2369 //luogo 5 41-50
2370 p41.setNome("Cecil");
2371 p41.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo","Nell''Armadiet
2372 to' non c'e nulla");
2373 p41.setLuogo(5);
2374 insPersonaggi->inserisci(41, p41);
2375 p42.setNome("Vivi");
2376 p42.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo","Nell''Armadiet
2377 to' non c'e nulla");
2378 p42.setOgg(true,54);
2379 p42.setFrasiconOgg1("Apri l''Armadietto', potresti trovare qualcosa di utile","La '
2380 Chiave' che hai preso non va bene per questo armadietto, cercane un'altra");
2381 p42.setLuogo(5);
2382 insPersonaggi->inserisci(42, p42);
2383 p43.setNome("Jin");
2384 p43.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo","Nell''Armadiet
2385 to' non c'e nulla");
2386 p43.setOgg(true,54);
2387 p43.setFrasiconOgg1("Togli la 'tata' prima di prendere il camice","Il 'camice' non
2388 serve a nulla, lascialo nell'armadietto");
2389 p43.setLuogo(5);
2390 insPersonaggi->inserisci(43, p43);

2391 //luogo 6 51-60
2392 p51.setNome("Cid");
2393 p51.setFrasi("Prendi il 'casco' Potrebbe tornarti utile","Non vedo nulla di utile q
2394 ui");
2395 p51.setOgg(true,50,51);
2396 p51.setFrasiconOgg1("Prendi la tuta dalla cabina di pilotaggio","La tuta non ti ser
2397 ve, vai direttamente all'esterno dell'astronave");
2398 p51.setFrasiconOgg2("Visto che hai la tuta, puoi uscire all'esterno dell'astronave"
2399 , "lascia il casco prima di uscire dall'astronave");
2400 p51.setLuogo(6);
2401 //modifiche D'Andria Dresda aggiunta dialogo a Cid
2402 Dialogo dialogocid=Dialogo("Cid");
2403 dialogocid=caricaDialoghiDaFile("Cid");
2404 p51.setDialogo(&dialogocid);
2405 //fine modifiche
2406 insPersonaggi->inserisci(51, p51);
2407 p52.setNome("Stainer");
2408 p52.setFrasi("Dopo aver preso il 'casco' cerca la 'tuta'","Ripostati un po sul lett
2409 o, potrebbe servirti a schiarire le idee");
2410 p52.setLuogo(6);
2411 insPersonaggi->inserisci(52, p52);
2412 p53.setNome("Bruce");
2413 p53.setFrasi("Non uscire all'esterno della nave senza 'casco' e 'tuta'","la 'tuta'
2414 e il 'casco' non servono per uscire all'esterno dell'astronave");
2415 p53.setLuogo(6);
2416 insPersonaggi->inserisci(53, p53);

2417 //luogo 7 61-70
2418 p61.setNome("Dom");
2419 p61.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senz
2420 a 'casco' altrimenti morirai");
2421 p61.setLuogo(7);
2422 insPersonaggi->inserisci(61, p61);
2423 p62.setNome("Gabriel");
2424 p62.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senz
2425 a 'casco' altrimenti morirai");
2426 p62.setOgg(true,50);
2427 p62.setFrasiconOgg1("Hai il il 'casco', ma ti manca la 'tuta' per poter uscire dal'
2428 astronave","hai il 'casco', premi rosso così potrai uscire dall'astronave");
2429 p62.setLuogo(7);
2430 insPersonaggi->inserisci(62, p62);
2431 p63.setNome("Martino");

```

```

2421     p63.setFrasi("non 'premere il rosso' senza avere il 'casco'", "'premi il rosso' senz
2422     a 'casco' altrimenti morirai");
2423     p63.setOgg(true,50,51);
2424     p63.setFrasiconOgg1("Hai il il 'casco', ma ti manca la 'tuta' per poter uscire dal'
2425     astronave", "hai il 'casco', premi rosso così potrai uscire dall'astronave");
2426     p63.setFrasiconOgg2("Hai sia il 'casco' che la 'tuta' ora puoi uscire dal'astronave"
2427     ", "lascia il 'casco', poi premi rosso così potrai uscire dall'astronave");
2428     p63.setLuogo(7);
2429     insPersonaggi->inserisci(63, p63);

2430     //luogo 8 71-80
2431     p71.setNome("Samuel");
2432     p71.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo b
2433     astasse fare il contrario?", "Il 'manuale' descrive come attivare il reattore, e se per
2434     spegnerlo bastasse fare la stessa procedura?");
2435     p71.setLuogo(8);
2436     insPersonaggi->inserisci(71, p71);
2437     p72.setNome("Shain");
2438     p72.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo b
2439     astasse fare il contrario?", "Il 'manuale' descrive come attivare il reattore, e se per
2440     spegnerlo bastasse fare la stessa procedura?");
2441     p72.setLuogo(8);
2442     insPersonaggi->inserisci(72, p72);
2443     p73.setNome("Zick");
2444     p73.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo b
2445     astasse fare il contrario?", "Il 'manuale' descrive come attivare il reattore, e se per
2446     spegnerlo bastasse fare la stessa procedura?");
2447     p73.setLuogo(8);
2448     insPersonaggi->inserisci(73, p73);

2449     //luogo 9 81-90
2450     p81.setNome("Lance");
2451     p81.setFrasi("Il 'secondo pilota' e' a nord", "si tolga il 'casco' e' sicuro qui");
2452
2453     p81.setLuogo(9);
2454     insPersonaggi->inserisci(81, p81);
2455     p82.setNome("Lauren");
2456     p82.setFrasi("Il 'secondo pilota' e' a nord", "si tolga il 'casco' e' sicuro qui");

2457     p82.setLuogo(9);
2458     insPersonaggi->inserisci(82, p82);
2459     p83.setNome("Lucas");
2460     p83.setFrasi("Il 'secondo pilota' e' a nord", "si tolga il 'casco' e' sicuro qui");

2461     p83.setLuogo(9);
2462     insPersonaggi->inserisci(83, p83);

2463     //luogo 10 91-100
2464     p91.setNome("Andre");
2465     p91.setFrasi("Soccorra il 'secondo pilota', e nei guai", "il 'secondo pilota' e' and
2466     ato perso nello spazio, e' inutile cercarlo");
2467     p91.setLuogo(10);
2468     insPersonaggi->inserisci(91, p91);
2469     p91.setNome("George");
2470     p91.setFrasi("Soccorra il 'secondo pilota', e nei guai", "il 'secondo pilota' e' and
2471     ato perso nello spazio, e' inutile cercarlo");
2472     p91.setLuogo(10);
2473     insPersonaggi->inserisci(92, p91);
2474     p91.setNome("Oscar");
2475     p91.setFrasi("Soccorra il 'secondo pilota', e nei guai", "il 'secondo pilota' e' and
2476     ato perso nello spazio, e' inutile cercarlo");
2477     p91.setLuogo(10);
2478     insPersonaggi->inserisci(93, p91);

2479     //aggiunta modifiche D'Andria Dresden al progetto di Federica Forte caricamento dial
2480     oghi da file
2481     scrivistatosufile("Cid",1);

```

```

2472     scrivistatosufile("Vincent",1);
2473     scrivistatosufile("Kail",1);
2474     //fine modifiche
2475 }
```

5.6.3.5 az_parla

```

2640 void Gioco::az_parla(Personaggi pg){
2641     int ris_controllo = 0;
2642     parola2[0]=toupper(parola2[0]);
2643     if(parola2==pg.getNome()){
2644         //modifiche effettuate da D'Andria Dresda per far dialogare i personaggi
2645         if(pg.getNome()=="Cid" || pg.getNome()=="Vincent" || pg.getNome()=="Kail"){
2646             int risposta;
2647             int statodialogo=caricastatodialogo(pg.getNome()); //leggo da file l'ultimo
2648             stato del dialogo
2649             pg.getDialogo()->setstato(statodialogo);
2650             pg.getDialogo()->visualizzadomande();
2651             bool dialogofinito=false;
2652             do{
2653                 cout << endl << "Digita numero risposta ----> ";
2654                 cin >> risposta;
2655                 cout << endl;
2656                 dialogofinito=pg.getDialogo()->rispondi(risposta);
2657                 if(risposta==0){
2658                     scrivistatosufile(pg.getNome(),pg.getDialogo()-
2659                         >getstato()); //in caso di risposta 0 si salva lo stato del dialogo, per riprenderlo da
2660                         //llo stesso punto la volta successiva che si incontrerà lo stesso personaggio
2661                 }
2662                 if (pg.getDialogo()->isFoglia()){
2663                     dialogofinito=true;
2664                     scrivistatosufile(pg.getNome(),1); //nel caso si raggiunga una foglia
2665                     //del dialogo si reimposta lo stato a 1
2666                 }
2667             } while(!dialogofinito);
2668         }
2669         else{
2670             if(pg.getIfNec()){
2671                 ris_controllo=controlla();
2672                 if(ris_controllo==0)
2673                     pg.SelezionaFrase2();
2674                 else if(ris_controllo==1)
2675                     pg.SelezionaFrase1();
2676                 else
2677                     parla(pg);
2678             }
2679         }
2680     }
2681     else
2682         cout << parola2 << " non e' in questa stanza" << endl;
2683 }
```

5.6.4 File aggiunti

5.6.4.1 AlberoNario

5.6.4.1.1 AlberoNario.h:

```

1 #ifndef ALBERONARIO_H
2 #define ALBERONARIO_H
```

```

3 #include <exception>
4 #include "BinAlbero.h"
5
6 /**
7 * Realizzazione dell'albero ennario attraverso l'albero binario.
8 * La memorizzazione dell'albero avviene nel seguente modo:
9 *   1. il PRIMOFIGLIO sarà FIGLIO SINISTRO.
10 *   2. il SUCCFRATELLO sarà FIGLIO DESTRO.
11 *
12 * author: Regina Zaccaria.
13 */
14
15 template <class tipoelem>
16 class Albero{
17 public:
18     //DEFINIZIONE DI NODO
19     typedef typename BinAlbero<tipoelem>::Nodo nodo;
20     //DEFINIZIONE DI NODO NULLO
21     static constexpr typename BinAlbero<tipoelem>::Nodo nil =BinAlbero<tipoelem>::nil;
22
23     //COSTRUTTORE
24     Albero();
25     //COSTRUTTORE DI COPIA
26     Albero(const Albero&);
27     //DISTRUTTORE
28     virtual ~Albero();
29     //METODO CHE CREA ALBERONARIO
30     void creaalbero();
31     //METODO CHE RESTITUISCE VERO SE L'ALBERO È VUOTO, FALSO ALTRIMENTI
32     bool alberovuoto() const;
33     //METODO CHE INSERISCE UN NODO NULLO COME RADICE
34     void insradice();
35     //METODO CHE RESTITUISCE IL NODO DELLA RADICE
36     nodo radice() const;
37     //METODO CHE RESTITUISCE IL NODO PADRE DEL NODO PASSATO
38     nodo padre(nodo);
39     //METODO CHE RESTITUISCE VERO SE IL NODO PASSATO È FOGLIA (QUINDI NON HA FIGLI), FA
40     LSO ALTRIMENTI
41     bool foglia(nodo) const;
42     //METODO CHE RESTITUISCE IL NODO PRIMOFIGLIO DI UN NODO PASSATO
43     nodo primofiglio(nodo) const;
44     //METODO CHE RESTITUISCE VERO SE IL NODO PASSATO NON HA FRATELLI SUCCESSIVI, FALSO
45     ALTRIMENTI
46     bool ultimofratello(nodo) const;
47     //METODO CHE RESTITUISCE IL NODO FRATELLO SUCCESSIVO DEL NODO PASSATO
48     nodo succfratello(nodo) const;
49     //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO C
50     OME NODO PRIMOFIGLIO DEL NODO PASSATO
51     void insprimosottoalbero(nodo, Albero&);
52     //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO C
53     OME UN NODO FRATELLO SUCCESSIVO DEL NODO PASSATO
54     void inssottoalbero(nodo, Albero&);
55     //METODO CHE PASSATO UN NODO CANCELLA LO STESSO E TUTTI I SUOI DISCENDENTI
56     void cancsottoalbero(nodo);
57     //METODO CHE SCRIVE L'ETICHETTA DI TIPO TIPOELEM ALL'INTERNO DEL NODO PASSATO
58     void scrivinodo(nodo&, tipoelem);
59     //METODO CHE RESTITUISCE L'ETICHETTA DEL NODO PASSATO
60     tipoelem legginodo(nodo) const;
61
62 private:
63     BinAlbero<tipoelem> tree;
64     //METODO CHE COPIA UN ALBERO PASSATO PARTENDO DAL NODO SINISTRO
65     void copia_albero_sx(const BinAlbero<tipoelem>&, const nodo&, nodo);
66     //METODO CHE COPIA L'ALBERO PASSATO PARTENDO DAL NODO DESTRO
67     void copia_albero_dx(const BinAlbero<tipoelem>&, const nodo&, nodo);
68 };

```

```

65
66 template <class tipoelem>
67 Albero<tipoelem>::Albero(){
68     creaalbero();
69 }
70
71 template <class tipoelem>
72 Albero<tipoelem>::Albero(const Albero& a){
73     if(!a.tree.binalberovuoto())
74     {
75         tree.insbinradice();
76         nodo n = tree.binradice();
77         nodo secondo_albero = a.tree.binradice();
78         tree.scrivinodo(n, a.tree.legginodo(secondo_albero));
79         if(a.tree.sinistruvuto(a.tree.binradice()))
80         {
81             if(!a.tree.destrovuoto(a.tree.binradice()))
82                 copia_albero_dx(a.tree,a.tree.figliodestro(a.tree.binradice()),tree
83 .binradice());
84             }
85             else
86                 copia_albero_sx(a.tree,a.tree.figliosinistro(a.tree.binradice()),tree.binra
87 dice());
88         }
89     }
90
91 template <class tipoelem>
92 Albero<tipoelem>::~Albero(){
93     //RICHIAAMA AUTOMATICAMENTE IL DISTRUTTORE
94     //DELL'ALBERO BINARIO
95 }
96
97 template <class tipoelem>
98 void Albero<tipoelem>::creaalbero(){
99     //RICHIAAMA AUTOMATICAMENTE IL COSTRUTTORE
100    //DELL'ALBERO BINARIO
101 }
102
103 template <class tipoelem>
104 bool Albero<tipoelem>::alberovuoto() const{
105     return tree.binalberovuoto();
106 }
107
108 template <class tipoelem>
109 void Albero<tipoelem>::insradice(){
110     tree.insbinradice();
111 }
112
113 template <class tipoelem>
114 typename Albero<tipoelem>::nodo Albero<tipoelem>::radice() const{
115     return tree.binradice();
116 }
117
118 template <class tipoelem>
119 typename Albero<tipoelem>::nodo Albero<tipoelem>::padre(typename Albero<tipoelem>::nodo
120 n){
121     return tree.binpadre(n);
122 }
123
124 template <class tipoelem>
125 bool Albero<tipoelem>::foglia(typename Albero<tipoelem>::nodo n) const{
126     if(tree.sinistruvuto(n))
127         return true;
128     else
129         return false;
130 }
```

```

129
130     template <class tipoelem>
131     typename Albero<tipoelem>::nodo Albero<tipoelem>::primofiglio(typename Albero<tipoelem>
132     ::nodo n) const{
133         return tree.figliosinistro(n);
134     }
135
136     template <class tipoelem>
137     bool Albero<tipoelem>::ultimofratello(typename Albero<tipoelem>::nodo n) const{
138         if(tree.destrovuto(n))
139             return true;
140         else
141             return false;
142     }
143
144     template <class tipoelem>
145     typename Albero<tipoelem>::nodo Albero<tipoelem>::succfratello(typename Albero<tipoelem>
146     ::nodo n) const{
147         return tree.figliodestro(n);
148     }
149
150     template<class tipoelem>
151     void Albero<tipoelem>::copia_albero_sx(const BinAlbero<tipoelem>& other, const nodo& ra
152     dice_sottoalbero, nodo nodo_padre)
153     {
154         tree.insfigliosinistro(nodo_padre);
155         nodo sx = tree.figliosinistro(nodo_padre);
156         tree.scrivinodo(sx, other.legginodo(radice_sottoalbero));
157
158         if(!other.sinistrovuoto(radice_sottoalbero))
159             copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), sx);
160         if(!other.destrovuto(radice_sottoalbero))
161             copia_albero_dx(other, other.figliodestro(radice_sottoalbero), sx);
162     }
163
164     template <class tipoelem>
165     void Albero<tipoelem>::copia_albero_dx(const BinAlbero<tipoelem>& other, const nodo& ra
166     dice_sottoalbero, nodo nodo_padre)
167     {
168         tree.insfigliodestro(nodo_padre);
169         nodo dx = tree.figliodestro(nodo_padre);
170         tree.scrivinodo(dx, other.legginodo(radice_sottoalbero));
171
172         if(!other.sinistrovuoto(radice_sottoalbero))
173             copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), dx);
174         if(!other.destrovuto(radice_sottoalbero))
175             copia_albero_dx(other, other.figliodestro(radice_sottoalbero), dx);
176     }
177
178     template <class tipoelem>
179     void Albero<tipoelem>::insprimosottoalbero(typename Albero<tipoelem>::nodo n, Albero& a
180     ){
181         nodo temp;
182         if(tree.sinistrovuoto(n)){
183             copia_albero_sx(a.tree,a.tree.binradice(),n);
184         }
185         else{
186             Albero<tipoelem> T;
187             T.insradice();
188             temp=T.radice();
189             T.scrivinodo(temp,tree.legginodo(tree.figliosinistro(n)));
190
191             if(tree.sinistrovuoto(tree.figliosinistro(n))){
192                 if(!tree.destrovuto(tree.figliosinistro(n))){
193                     copia_albero_dx(tree,tree.figliodestro(tree.figliosinistro(n)),temp
194                 );
195             }
196         }
197     }

```

```

190     }
191     else{
192         copia_albero_sx(tree,tree.figliosinistro(tree.figliosinistro(n)),temp);
193     }
194
195     tree.cancsottobinalbero(tree.figliosinistro(n));
196
197     copia_albero_sx(a.tree,a.tree.binradice(),n);
198     temp=tree.figliosinistro(n);
199     copia_albero_dx(T.tree,T.radice(),temp);
200 }
201
202
203
204 template <class tipoelem>
205 void Albero<tipoelem>::inssottoalbero(typename Albero<tipoelem>::nodo n, Albero& a){
206     /* L'albero è ottenuto aggiungendo il sottoalbero a di radice r dove r diventa il fratello successivo
207     di n. n non è la radice*/
208
209     nodo temp;
210     if(n!=tree.binradice()){
211         if(tree.destruvoto(n)){
212             copia_albero_dx(a.tree,a.radice(),n);
213         }
214         else{
215             Albero<tipoelem> T;
216             T.insradice();
217
218             temp=T.radice();
219             T.scrivinodo(temp,tree.legginodo(tree.figliodestro(n)));
220
221             if(tree.sinistruvoto(tree.figliodestro(n))){
222                 if(!tree.destruvoto(tree.figliodestro(n))){
223                     copia_albero_dx(tree,tree.figliodestro(tree.figliodestro(n))
224                     ,temp);
225                 }
226                 else{
227                     copia_albero_sx(tree,tree.figliosinistro(tree.figliodestro(n)),temp
228                     );
229                 }
230
231                 cancottoalbero(tree.figliodestro(n));
232
233                 copia_albero_dx(a.tree,a.radice(),n);
234                 temp=tree.figliodestro(n);
235                 copia_albero_dx(T.tree,T.radice(),temp);
236             }
237         }
238
239     template <class tipoelem>
240     void Albero<tipoelem>::cancsottoalbero(typename Albero<tipoelem>::nodo n){
241         /* L'albero è ottenuto togliendo il sottoalbero di radice n e tutti i suoi discendenti*/
242         tree.cancsottobinalbero(n);
243     }
244
245     template <class tipoelem>
246     void Albero<tipoelem>::scrivinodo(typename Albero<tipoelem>::nodo& n, tipoelem elem){
247         tree.scrivinodo(n,elem);
248     }
249
250     template <class tipoelem>
251     tipoelem Albero<tipoelem>::legginodo(typename Albero<tipoelem>::nodo n) const{
252         return tree.legginodo(n);

```

```
253 }
254 #endif // ALBERONARIO_H
```

5.6.4.1.2 BinAlbero.h

```
1 #ifndef BINALBERI_H_INCLUDED
2 #define BINALBERI_H_INCLUDED
3 #include "Nodo_Albero_Binario.h"
4
5 /**
6 * Realizzazione dell'ALBERO BINARIO totalmente dinamica.
7 * Riferimento al padre, al figlio sinistro e al figlio destro di un nodo.
8 * author: Regina Zaccaria.
9 */
10 template <class TIPOETICHETTA>
11 class BinAlbero{
12
13     public:
14         //DEFINIZIONE DEL NODO
15         typedef Cella_Binalbero<TIPOETICHETTA>* Nodo;
16         //DICHIARAZIONE NODO NULLO
17         static constexpr Nodo nil=nullptr;
18         //COSTRUTTORE
19         BinAlbero();
20         //DISTRUTTORE
21         ~BinAlbero();
22         //METODO CHE CREA UN ALBERO BINARIO
23         void creabinalbero();
24         //METODO CHE RESTITUISCE VERO SE L'ABERO BINARIO È VUOTO, FALSO ALTRIMENTI
25         bool binalberovuoto()const;
26         //METODO CHE RESTITUISCE LA RADICE DELL'ALBERO BINARIO
27         Nodo binradice()const;
28         //METODO CHE PASSATO UN NODO RESTITUISCE IL SUO PADRE
29         Nodo binpadre(Nodo)const;
30         //METODO CHE RESTITUISCE VERO SE IL FIGLIO SINISTRO DEL NODO PASSATO NON ESISTE,
31         //FALSO ALTRIMENTI
32         bool sinistrovuoto(Nodo)const;
33         //METODO CHE RESTITUISCE VERO SE IL FIGLIO DESTRO DEL NODO PASSATO NON ESISTE, F
34         //ALSO ALTRIMENTI
35         bool destrovuoto(Nodo)const;
36         //METODO CHE RESTITUISCE IL FIGLIO SINISTRO DEL NODO PASSATO
37         Nodo figliosinistro(Nodo)const;
38         //METODO CHE RESTITUISCE IL FIGLIO DESTRO DEL NODO PASSATO
39         Nodo figliodestro(Nodo)const;
40         //METODO CHE DATI DUE ALBERI, CREA UNA RADICE NULLA ALL'ALBERO BINARIO IMPLICITO
41         ,
42         //INSERISCE COME FIGLIO SINISTRO LA RADICE DEL PRIMO ALBERO PASSATO E COPIATI I
43         //SUOI DISCENDENTI,
44         //E INSERISCE COME FIGLIO DESTRO IL SECONDO ALBERO PASSATO E COPIATI I SUOI DISC
45         //ENDENTI.
46         void costrbinalbero(BinAlbero<TIPOETICHETTA>&,BinAlbero<TIPOETICHETTA>&);
47         //METODO CHE PASSATO UN NODO LO CANCELLA E CANCELLA TUTTI I SUOI DISCENDENTI
48         void cancottobinalbero(Nodo);
49         //METODO CHE PASSATO UN NODO RESTITUISCE LA SUA ETICHETTA
50         TIPOETICHETTA legginodo(Nodo)const;
51         //METODO CHE PASSATO UN NODO SCRIVE AL SUO INTERNO L'ETICHETTA DI TIPO TIPOLEM
52
53         void scrivinodo(Nodo, TIPOETICHETTA);
54         //METODO CHE INSERISCE LA RADICE NULLA
55         void insbinradice();
56         //METODO CHE INSERISCE COME FIGLIO SINISTRO IL NODO PASSATO
57         void insfigliosinistro(Nodo);
58         //METODO CHE INSERISCE COME FIGLIO DESTRO IL NODO PASSATO
59         void insfigliodestro(Nodo);

60     private:
61         Nodo radice;
```

```

57    };
58
59
60    template <class TIPOETICHETTA>
61    BinAlbero<TIPOETICHETTA>::BinAlbero(){
62        creabinalbero();
63    }
64
65    template <class TIPOETICHETTA>
66    BinAlbero<TIPOETICHETTA>::~BinAlbero(){
67        /* if(radice!=nil)
68            cancottobinalbero(binradice());
69        */
70    }
71
72
73    template <class TIPOETICHETTA>
74    void BinAlbero<TIPOETICHETTA>::creabinalbero(){
75        radice=nil;
76    }
77
78    template <class TIPOETICHETTA>
79    bool BinAlbero<TIPOETICHETTA>::binalberovuoto()const{
80        if(radice==nil)
81            return true;
82        else
83            return false;
84    }
85
86    template <class TIPOETICHETTA>
87    typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binradice()const{
88
89        return radice;
90    }
91
92    template <class TIPOETICHETTA>
93        typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binpadre(Nodo pad
94 re)const{
95        return padre->getPadre();
96    }
97
98    template <class TIPOETICHETTA>
99        bool BinAlbero<TIPOETICHETTA>::sinistrovuoto(Nodo sx)const{
100            if(sx->getFiglioSx()==nil)
101                return true;
102            else
103                return false;
104        }
105
106        template <class TIPOETICHETTA>
107            bool BinAlbero<TIPOETICHETTA>::destrovuoto(Nodo dx)const{
108                if(dx->getFiglioDx()==nil)
109                    return true;
110                else
111                    return false;
112        }
113
114        template <class TIPOETICHETTA>
115            typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliosinistro(No
116 do sx)const{
117            return sx->getFiglioSx();
118        }
119
120        template <class TIPOETICHETTA>
121            typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliodestro(No
122 do dx)const{
123            return dx->getFiglioDx();

```

```

120     }
121
122     template <class TIPOETICHETTA>
123         void BinAlbero<TIPOETICHETTA>::costrbinalbero(BinAlbero<TIPOETICHETTA> &A,BinAlbero
124 <TIPOETICHETTA> &B){
125         radice=new Nodo;
126         radice->setPadre(nil);
127
128         if(!A.binalberovuoto()){
129             radice->setFiglioSx(A.binradice());
130             radice->getFiglioSx()->setPadre(radice);
131         }
132         else
133             radice->setFiglioSx(nil);
134
135         if(!B.binalberovuoto()){
136             radice->setFiglioDx(B.binradice());
137             radice->getFiglioDx()->setPadre(radice);
138         }
139         else
140             radice->setFiglioDx(nil);
141     }
142
143     template <class TIPOETICHETTA>
144         void BinAlbero<TIPOETICHETTA>::cancsottobinalbero(Nodo r){
145
146         if(!sinistrovuoto(r))
147             cancsottobinalbero(r->getFiglioSx());
148
149         if(!destrovuoto(r))
150             cancsottobinalbero(r->getFiglioDx());
151
152         if(radice!=r)
153         {
154             Nodo temp;
155             temp=binpadre(r);
156
157             if(temp->getFiglioSx()==r)
158                 temp->setFiglioSx(nil);
159             else
160                 temp->setFiglioDx(nil);
161         }
162         else
163             radice=nil;
164
165         delete r;
166     }
167
168     template <class TIPOETICHETTA>
169         TIPOETICHETTA BinAlbero<TIPOETICHETTA>::legginodo(Nodo n) const{
170             return n->getEtichetta();
171         }
172
173
174     template <class TIPOETICHETTA>
175         void BinAlbero<TIPOETICHETTA>::scrivinodo(Nodo n, TIPOETICHETTA e){
176             n->setEtichetta(e);
177         }
178
179     template <class TIPOETICHETTA>
180         void BinAlbero<TIPOETICHETTA>::insbinradice(){
181             radice= new Cella_Binalbero<TIPOETICHETTA>;
182             radice->setFiglioDx(nil);
183             radice->setFiglioSx(nil);
184             radice->setPadre(nil);
185         }

```

```

186 template <class TIPOETICHETTA>
187     void BinAlbero<TIPOETICHETTA>::insfigliosinistro(Nodo n){
188         Nodo temp = new Cella_Binalbero<TIPOETICHETTA>;
189         n->setFiglioSx(temp);
190         temp->setPadre(n);
191         temp->setFiglioDx(nil);
192         temp->setFiglioSx(nil);
193     }
194
195 template <class TIPOETICHETTA>
196     void BinAlbero<TIPOETICHETTA>::insfigliodestro(Nodo n){
197         Nodo temp = new Cella_Binalbero<TIPOETICHETTA>;
198         n->setFiglioDx(temp);
199         temp->setPadre(n);
200         temp->setFiglioDx(nil);
201         temp->setFiglioSx(nil);
202     }
203
204
205 #endif // BINALBERI_H_INCLUDED

```

5.6.4.1.3 Nodo_Albero_Binario.h

```

1 #ifndef NODO_ALBERO_BINARIO_H_INCLUDED
2 #define NODO_ALBERO_BINARIO_H_INCLUDED
3
4 /**
5 * Realizzazione del Nodo dell'albero binario.
6 * Il nodo conterrà riferimento al FIGLIO SINISTRO,
7 * riferimento al FIGLIO DESTRO e riferimento al PADRE.
8 * Inoltre avrà un campo ETICHETTA di tipo TIPOETICHETTA.
9 */
10 template <class TIPOETICHETTA>
11     class Cella_Binalbero{
12     public:
13         //COSTRUTTORE
14         Cella_Binalbero();
15         //DISTRUTTORE
16         ~Cella_Binalbero();
17         //METODO CHE SETTA IL FIGLIO SINISTRO
18         void setFiglioSx(Cella_Binalbero<TIPOETICHETTA*>*);
19         //METODO CHE SETTA IL FIGLIO DESTRO
20         void setFiglioDx(Cella_Binalbero<TIPOETICHETTA*>*);
21         //METODO CHE SETTA IL PADRE
22         void setPadre(Cella_Binalbero<TIPOETICHETTA*>*);
23         //METODO CHE SETTA L'ETICHETTA
24         void setEtichetta(TIPOETICHETTA&);
25         //METODO CHE RESTITUISCE IL RIFERIMENTO AL FIGLIO SINISTRO
26         Cella_Binalbero<TIPOETICHETTA*>* getFiglioSx();
27         //METODO CHE RESTITUISCE IL RIFERIMENTO AL FIGLIO DESTRO
28         Cella_Binalbero<TIPOETICHETTA*>* getFiglioDx();
29         //METODO CHE RESTITUISCE IL RIFERIMENTO AL PADRE
30         Cella_Binalbero<TIPOETICHETTA*>* getPadre();
31         //METODO CHE RESTITUISCE L'ETICHETTA
32         TIPOETICHETTA getEtichetta();
33         //OVERLOAD DELL'OPERATORE ==
34         bool operator == (Cella_Binalbero<TIPOETICHETTA>);
35         //COSTRUTTORE DI COPIA
36         Cella_Binalbero(const Cella_Binalbero& );
37         //OVERLOAD DELL'OPERATORE =
38         void operator=(const Cella_Binalbero & );
39
40     private:
        Cella_Binalbero<TIPOETICHETTA*>* FiglioDx;

```

```

41         Cella_Binalbero<TIPOETICHETTA>* FiglioSx;
42         Cella_Binalbero<TIPOETICHETTA>* Padre;
43         TIPOETICHETTA etichetta;
44     };
45
46
47     template <class TIPOETICHETTA>
48         Cella_Binalbero<TIPOETICHETTA>::Cella_Binalbero(){
49             FiglioDx=nullptr;
50             FiglioSx=nullptr;
51             Padre=nullptr;
52         }
53
54     template <class TIPOETICHETTA>
55         Cella_Binalbero<TIPOETICHETTA>::~Cella_Binalbero(){
56     }
57
58     template <class TIPOETICHETTA>
59         void Cella_Binalbero<TIPOETICHETTA>::setFiglioSx(Cella_Binalbero<TIPOETICHETTA>* sx
60     ){
61         FiglioSx=sx;
62     }
63
64     template <class TIPOETICHETTA>
65         void Cella_Binalbero<TIPOETICHETTA>::setFiglioDx(Cella_Binalbero<TIPOETICHETTA>* dx
66     ){
67         FiglioDx=dx;
68
69     template <class TIPOETICHETTA>
70         void Cella_Binalbero<TIPOETICHETTA>::setPadre(Cella_Binalbero<TIPOETICHETTA>* p){
71             Padre=p;
72         }
73
74     template <class TIPOETICHETTA>
75         void Cella_Binalbero<TIPOETICHETTA>::setEtichetta(TIPOETICHETTA& e){
76             etichetta=e;
77         }
78
79     template <class TIPOETICHETTA>
80         Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getFiglioSx(){
81             return FiglioSx;
82         }
83
84     template <class TIPOETICHETTA>
85         Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getFiglioDx(){
86             return FiglioDx;
87         }
88
89     template <class TIPOETICHETTA>
90         Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getPadre(){
91             return Padre;
92         }
93
94     template <class TIPOETICHETTA>
95         TIPOETICHETTA Cella_Binalbero<TIPOETICHETTA>::getEtichetta(){
96             return etichetta;
97         }
98
99     template <class TIPOETICHETTA>
100        bool Cella_Binalbero<TIPOETICHETTA>::operator == (Cella_Binalbero<TIPOETICHETTA> b
101    ){
102        if(getEtichetta()==b.getEtichetta())
103            return true;
104        else
105            return false;
106    }

```

```

105     template <class TIPOETICHETTA>
106     Cella_Binalbero<TIPOETICHETTA>::Cella_Binalbero(const Cella_Binalbero& c){
107         etichetta=c.getEtichetta();
108         FiglioDx=c.getFiglioDx();
109         FiglioSx=c.getFiglioSx();
110         Padre=c.getPadre();
111     }
112
113     template <class TIPOETICHETTA>
114     void Cella_Binalbero<TIPOETICHETTA>::operator=(const Cella_Binalbero& c ){
115         etichetta=c.getEtichetta();
116         Padre=c.getPadre();
117         FiglioDx=c.getFiglioDx();
118         FiglioSx=c.getFiglioSx();
119     }
120 #endif // NODO_ALBERO_BINARIO_H_INCLUDED

```

5.6.4.2 Domanda

5.6.4.2.1 Domanda.h:

```

1     #ifndef DOMANDA_H
2     #define DOMANDA_H
3     #include "string"
4     using namespace std;
5
6     class Domanda{
7     public:
8         Domanda();
9         Domanda(const Domanda& ); //costruttore di copia
10        virtual ~Domanda();
11        Domanda(int, string);//costruttore codice-domanda
12        int leggicodice();//funzione che ritorna il codice della domanda
13        string leggidomanda();//funzione che ritorna la stringa contenente la domanda con le relative possibilità di risposta
14
15     private:
16         string domanda;
17         int codicedomanda;
18     };
19
20
21
22 #endif // DOMANDA_H

```

5.6.4.2.2 Domanda.cpp:

```

1     #include "Domanda.h"
2
3     Domanda::Domanda() {}
4
5     //costruttore di copia
6     Domanda::Domanda(const Domanda& d){
7         domanda = d.domanda;
8         codicedomanda = d.codicedomanda;
9     }
10
11    Domanda::~Domanda() {}
12
13    Domanda::Domanda(int codice, string stringa){
14        this->codicedomanda=codice;
15        this->domanda=stringa;
16    }
17
18    int Domanda::leggicodice(){
19        return this->codicedomanda;

```

```

20     }
21
22     string Domanda::leggidomanda(){
23         return this->domanda;
24     }

```

5.6.4.3 Dialogo

5.6.4.3.1 Dialogo.h:

```

1 #ifndef DIALOGO_H
2 #define DIALOGO_H
3 #include "AlberoNario.h"
4 #include "Domanda.h"
5 #include "math.h"
6 #include <iostream>
7 #include "Pila.h"
8 using namespace std;
9 class Dialogo{
10 public:
11     Dialogo();
12     Dialogo(string);
13     Dialogo(const Dialogo& );
14     virtual ~Dialogo();
15     //funzione ricorsiva che fa ritornare il nodo domanda corrispondente al codice richiesto
16     Albero<Domanda>::nodo trovanodo(Albero<Domanda>::nodo, int);
17
18     string leggiPersonaggio();
19     int getstato();
20     void setstato(int);
21
22     //metodi usati per costruire l'albero dialogo
23     void inseriscidomanda(int,string);
24     void inseriscipresentazione(string);
25     void inserisciprimaopzione(Albero<Domanda>::nodo,int, string);
26     void inserisciopzionessuccessiva(Albero<Domanda>::nodo,int, string);
27
28     //metodo che visualizza a video la domanda allo stato corrente
29     void visualizzadomande();
30     // metodo che cambia lo stato del dialogo in base alla risposta data
31     void cambiastato(int);
32     //metodo che riceve la risposta e controlla la sua validità
33     bool rispondi(int);
34     //funzione che ritorna true quando il dialogo è vuoto
35     bool dialogoVuoto();
36     //funzione che ritorna true quando si raggiunge una foglia dell'albero del dialogo
37
38     bool isFoglia();
39
40 private:
41     string personaggio;//nome personaggio a cui è associato il dialogo
42     int stato;//stato attuale del dialogo
43     Albero<Domanda> dialogo;//albero contenente le domande del dialogo
44 };
45 #endif // DIALOGO_H

```

5.6.4.3.2 Dialogo.cpp:

```

1 #include "Dialogo.h"
2
3 Dialogo::Dialogo() {}
4 Dialogo::~Dialogo() {
5
}

```

```

6 Dialogo::Dialogo(const Dialogo& d){
7     personaggio = d.personaggio;
8     stato = d.stato;
9     dialogo = Albero<Domanda>(d.dialogo);
10    }
11
12 Dialogo::Dialogo(string nomepersonaggio){
13     this->personaggio=nomepersonaggio;
14     this->stato=1;
15 }
16
17 void Dialogo::setstato(int numero){
18     this->stato=numero;
19 }
20
21 int Dialogo::getstato(){
22     return (this->stato);
23 }
24 string Dialogo::leggiPersonaggio(){
25     return (this->personaggio);
26 }
27
28 void Dialogo::inseriscidomanda(int codice, string frase){
29     if (codice==1)
30         inseriscipresentazione(frase);
31     else
32     {
33         Albero<Domanda>::nodo indicenodo=Albero<Domanda>::nil;
34         int codicedacercare=0;
35         int modulocodice=codice%10;
36         if(modulocodice==1){
37             codicedacercare=(codice-1)/10;
38             indicenodo = trovanodo(dialogo.radice(),codicedacercare);
39             inserisciprimaopzione(indicenodo,codice,frase);
40         }
41         else{
42             codicedacercare=codice-1;
43             indicenodo = trovanodo(dialogo.radice(),codicedacercare);
44             inserisciopzionessuccessiva(indicenodo,codice,frase);
45         }
46     }
47 }
48
49 Albero<Domanda>::nodo Dialogo::trovanodo(Albero<Domanda>::nodo indice, int codice){
50     Albero<Domanda>::nodo nodo;
51     Albero<Domanda>::nodo nododaritornare=Albero<Domanda>::nil;
52
53     if(codice==dialogo.legginodo(indice).leggicodice())
54         nododaritornare=indice;
55
56     else{
57         if (!dialogo.foglia(indice)){
58             nodo=dialogo.primofiglio(indice);
59             while (!dialogo.ultimofratello(nodo) && nododaritornare==Albero<Domanda>::ni
60             1){
61                 nododaritornare=trovanodo(nodo,codice);
62                 nodo=dialogo.succfratello(nodo);
63             }
64             if (nododaritornare==Albero<Domanda>::nil)
65                 nododaritornare=trovanodo(nodo,codice);
66         }
67     }
68     return nododaritornare;
69 }
70
71

```

```

72 void Dialogo::inseriscipresentazione(string presentazione){
73     Domanda saluto = Domanda(1,presentazione);
74     Albero<Domanda>::nodo nodosaluto;
75     dialogo.insradice();
76     nodosaluto=dialogo.radice();
77     dialogo.scrivinodo(nodosaluto,saluto);
78 }
79
80 void Dialogo::inserisciprimaopzione(Albero<Domanda>::nodo domandacorrente,int codice, s
81     tring frase){
82     Albero<Domanda> alberoopzione;
83     Albero<Domanda>::nodo nodoopzione;
84     Domanda nuovadomanda= Domanda(codice,frase);
85     alberoopzione.insradice();
86     nodoopzione=alberoopzione.radice();
87     alberoopzione.scrivinodo(nodoopzione, nuovadomanda);
88     dialogo.insprimosottoalbero(domandacorrente,alberoopzione);
89 }
90
91 void Dialogo::inserisciopzionessuccessiva(Albero<Domanda>::nodo nodoprecedente,int codic
92 e, string frase){
93     Albero<Domanda> alberoopzione;
94     Albero<Domanda>::nodo nodoopzione;
95     Domanda nuovadomanda= Domanda(codice,frase);
96     alberoopzione.insradice();
97     nodoopzione=alberoopzione.radice();
98     alberoopzione.scrivinodo(nodoopzione, nuovadomanda);
99     dialogo.inssottoalbero(nodoprecedente,alberoopzione);
100 }
101
102 void Dialogo::visualizzadomande(){
103     int domandadavisualizzare=this->stato;
104
105     Albero<Domanda>::nodo indicenodo=Albero<Domanda>::nil;
106     indicenodo = trovanodo(dialogo.radice(),domandadavisualizzare);
107     cout<<dialogo.legginodo(indicenodo).leggidomanda();
108 }
109
110 void Dialogo::cambiastato(int risposta){
111     int statosuccessivo=(this->stato)*10+risposta;
112     Albero<Domanda>::nodo nodosuccessivo=Albero<Domanda>::nil;
113     nodosuccessivo = trovanodo(dialogo.radice(),statosuccessivo);
114     if(nodosuccessivo!=Albero<Domanda>::nil)
115         this->stato=statosuccessivo;
116     else
117         cout<< "Risposta non valida\n";
118     visualizzadomande();
119 }
120
121 bool Dialogo::rispondi(int risposta){
122     if (risposta==0)
123         return true;
124     else{
125         cambiastato(risposta);
126         return false;
127     }
128 }
129
130 bool Dialogo::dialogoVuoto(){
131     return(dialogo.alberovuoto());
132 }
133
134 bool Dialogo::isFoglia(){
135     int domandadavisualizzare=this->stato;
136     Albero<Domanda>::nodo indicenodo=Albero<Domanda>::nil;
137     indicenodo = trovanodo(dialogo.radice(),domandadavisualizzare);
138     return(dialogo.foglia(indicenodo));

```

5.6.4.3.3 Dialoghi.txt

```

-Cid|1*Ciao, sono Cid!!!$Credi in Dio?$1:si$2:no$0:interrompi dialogo$#
-
Cid|11*Benissimo, sei gia' andato in chiesa?$$1:Si ci sono stato$2:Non ancora$0:interrompi dialogo$#
-
Cid|12*Male, molto male. Solo Lui puo' aiutarti!$#
-Cid|111*Bene, ti sei confessato?$$1:Si$2:no$0:interrompi dialogo$#
-Cid|112*Vai finche' sei tempo e purifica la tua anima confessandoti!$#
-Cid|1111*Bravissimo!! Dio ti condurrà verso la salvezza!$#
-Cid|1112*Tornaci e confessati! Che Dio ti benedica!!$#
-
Vincent|1*Ciao, sono Vincent!$$Preferisci il fast food oppure il ristorante?$$1:Fast food$2:$Ristorante$0:Interrompi dialogo$#
-
Vincent|11*Interessante, ci sei gia' stato?$$1:Si$2:Non ancora$0:Interrompi dialogo$#
-
Vincent|12*Benissimo, ti consiglio di andarci per gustare delle specialita' galattiche!$Così ti piace mangiare?$$1:Carne$2:Pesce$3:Verdure$0:Interrompi dialogo$#
-
Vincent|111*Grandioso, cosa ne pensi del fast food?$$1:E' un buon posto e si mangia bene$2:Non ci andro' mai piu'$3:Mi sono trovato lì di passaggio e non ho mangiato nulla$0:Interrompi dialogo$#
-
Vincent|112*Cosa aspetti?! Vai, prima che tu muoia di fame!!!$#
-
Vincent|121*Ottimo, credo che il loro arrosto sia uno dei piu' buoni in tutto l'universo!$#
-
Vincent|122*Hai gia' provato il loro misto di frutti di mare?$$1:Si$2:No$0:interrompi dialogo$#
-
Vincent|123*Prova gli spinaci alla panna, sono deliziosi$#
-
Vincent|1111*Gia', lo penso anche io, prova anche il ristorante, l'ho trovato adorabile.$#
-
Vincent|1112*Mi dispiace, vorra' dire che andrai al ristorante per mangiare qualcosa.$#
-
Vincent|1113*Male, prova il loro gelato alla panna e' delizioso, ricordati di usare il cucchiaio!$#
-
Vincent|1221*Io l'ho apprezzato molto!$#
-
Vincent|1222*Provalo, non te ne pentirai!$#
-
Kail|1*Ciao, sono Kail!$$Hai bisogno di cure?$$1:Si, grazie$2:No, grazie$0:interrompi dialogo$#
-
Kail|11*Spero nulla di grave, hai gia' preso la tessera sanitaria?$$1:Si$2:No$0:interrompi dialogo$#
-
Kail|12*Apprezzo il fatto che tu stia bene, se dovesse servirti recati al pronto soccorso.$#
-
Kail|111*Benissimo, corri al pronto soccorso e usa il terminale!$#
-
Kail|112*Cosa aspetti?! Hai almeno idea di dove si trovi?$$1:Si, so dov'e'$2:Non ne ho la più pallida idea$0:interrompi dialogo$#
-
Kail|1121*Ottimo, corri a prenderla e recati al pronto soccorso!!!$#
-
Kail|1122*Come hai fatto a non vederla?! E' nella tua cabina! Prendila e vai al pronto soccorso!!!$#
?

```

5.6.4.3.4 Statidialoghi.txt

?

5.7 SALA SCOMMESSE E SIMULATORE

5.7.1 Implementazione luogo Sala scommesse e nuove azioni nel gioco

Come precedentemente spiegato, è stato aggiunto un nuovo luogo denominato "Sala scommesse" all'interno della mappa di gioco.

Per effettuare la sua aggiunta si è dovuto:

- Cambiare il numero di luoghi presenti nel file mappa.nav, modificando da 22 a 23.
- Poiché il luogo sala scommesse si trova, partendo dalla cabina iniziale, andando a nord del corridoio, scendendo, ed andando al nord della banca, è stato modificato il codice di movimento di quest'ultima:
 - 15,In una banca,230000140000,via 6,2,2
 - È stato aggiunto il nuovo luogo "Sala scommesse":
 - 23,Nella Sala Scommessa,001500000000, via 6,2,2

Per aggiungere le nuove funzionalità offerte dal nuovo luogo "Sala scommesse", sono state aggiunte due nuove azioni all'interno del file Astro.h

void azione_80();

void azione_81();

Di seguito si riporta l'implementazione di tali azioni nel file Astro.cpp

```
void Astro::azione_80() //Gioca al simulatore oppure avvia simulatore
{
    if(portafoglio.hai_Portafoglio(oggetti))
    {

//Aggiunta coda random
    int generatore_coda; //Variabile di selezione e di generatore coda
    string input_utente; //Stringa contenente l'input dell'utente
    bool conferma=false; //Conferma utente
    bool controllo=false; //Variabile controllo
    float puntata=0; //Variabile contenente la somma puntata dal personaggio
    float saldo=0; //variabile d'appoggio contenente il saldo del personaggio
    generatore_coda=rand()%6; //Genera una variabile casuale compresa tra 0 e 5
    if(generatore_coda>0) //Se ci sono persone in coda, chiedi al Giocatore cosa
desidera fare
    {
        interfaccia.scrivi_parziale(generatore_coda);
        interfaccia.scrivi_parziale(" attendono per usare il simulatore.Desideri
attendere che si liberi? [si/no] ");
        while(controllo!=true)
        {
            cin>>input_utente;
```

```

        if(input_utente=="S" || input_utente=="s" || input_utente=="Si" ||input_utente=="SI" ||input_utente=="si") //Se l'utente digita si, aggiorna conferma
        {
            conferma=true; //l'utente conferma di voler utilizzare il simulatore
            controllo=true; //Aggiornamento controllo
            tempo=generatore_coda; //Diminuisce il tempo di gioco a seconda
            del numero di persone presenti in coda
        }
        else if(input_utente=="N" || input_utente=="n" || input_utente=="No" ||input_utente=="NO" ||input_utente=="no") //Se l'utente digita no,aggiorna conferma
        {
            conferma=false; //L'utente ha confermato di non voler utilizzare il
            terminale
            controllo=true; //Aggiorna il controllo
        }
        else
        {
            interfaccia.scrivi("Non capisco.");
            aggiorna_tempo();
            interfaccia.a_capo();
            interfaccia.scrivi_parziale(generatore_coda);
            interfaccia.scrivi_parziale(" attendono per usare il
            simulatore.Desideri attendere che si liberi? [si/no] ");
            interfaccia.a_capo();
        }
    }
}
else
{
    conferma=true; //Se nella fila non sono presenti persone , usa il simulatore
    senza bisogno di fare la fila
}
if(conferma==true)
{
    Sleep(850);
    interfaccia.a_capo();
    interfaccia.scrivi("####BENVENUTO NEL SIMULATORE####");
    interfaccia.scrivi("Le regole sono semplicissime!");
    interfaccia.scrivi("Scegli l'astronave su cui puntare!");
    interfaccia.scrivi("Se risulterà vincente potrai triplicare la somma puntata!!");
    do
    {
        interfaccia.a_capo();

```

```

int astronave,puntata, scelta_ut;
interfaccia.scrivi_parziale("Credito disponibile ");
interfaccia.scrivi_parziale(portafoglio.get_contanti());
interfaccia.scrivi_parziale(" Euro ");
interfaccia.a_capo();
interfaccia.scrivi("INSERISCI LA CIFRA DA PUNTARE");
interfaccia.scrivi_parziale("Euro: ");
cin>>puntata;//acquisizione somma da puntare

if(portafoglio.get_contanti()>=puntata && puntata!=0)
{
    saldo=portafoglio.get_contanti();
    saldo=saldo-puntata;
    portafoglio.set_contanti(saldo);//aggiornamento saldo
    do
    {
        interfaccia.a_capo();
        interfaccia.a_capo();
        interfaccia.scrivi("Ecco la lista delle astronavi in gara:");
        interfaccia.scrivi("ASTRONAVE 1 :Leo      ASTRONAVE
2:Mike ");
        interfaccia.scrivi("ASTRONAVE 3 :Amy      ASTRONAVE
4:Paul ");
        interfaccia.scrivi("ASTRONAVE 5 :Harrison   ASTRONAVE
6:John ");
        interfaccia.a_capo();
        interfaccia.scrivi("Inserisci il numero dell'astronave su cui
puntare");
        cin>>scelta_ut;//acquisizione della scelta dell'utente riguardante il
numero dell'astronave
    }
    while(scelta_ut<0 || scelta_ut>6);
    astronave=rand()%6+1;//generazione di una variabile casuale da 1 a 6
    if(astronave==scelta_ut)//controlla se la scelta inserita dall'utente è
uguale al numero dell'astronave vincitrice
    {
        interfaccia.scrivi_parziale("COMPLIMENTI HAI VINTO ");
    }
}

```

```

        interfaccia.scrivi_parziale(puntata*3);
        interfaccia.scrivi_parziale(" EURO");
        interfaccia.a_capo();
        interfaccia.a_capo();
        saldo=saldo+(puntata*3);
        portafoglio.set_contanti(saldo);
    }
else
{
    interfaccia.a_capo();
    interfaccia.scrivi("Peccato,hai perso!!!");
    interfaccia.scrivi_parziale("L'astronave vincitrice e' stata la numero
");
    interfaccia.scrivi_parziale(astronave);
    interfaccia.a_capo();

}
}
else
{
    interfaccia.a_capo();
    interfaccia.scrivi("Non hai credito sufficiente!");
    interfaccia.a_capo();
    interfaccia.a_capo();
    Sleep(850); //Attendi alcuni istanti

}
interfaccia.a_capo();
tempo--;
aggiorna_tempo();
interfaccia.scrivi("VUOI GIOCARE ANCORA?");
cin>>input_utente;//acquisizione scelta dell'utente
}while(input_utente=="S" || input_utente=="s" || input_utente=="Si" ||input_uten
te=="SI" ||input_utente=="si");
}
else
{
    interfaccia.scrivi("Torna a giocare con il Simulatore :D");
}
}
else
{
    interfaccia.scrivi("Hai bisogno del portafoglio");
}

```

```

        }
        storia_gioco.insStoria(stringa_comando , "hai giocato al simulatore di corse di
astronavi");
    }
}

```

```

void Astro :: azione_81 () //Guarda simulatore
{
    interfaccia. scrivi ( "L'insegna dice: PIAZZA LA TUA SCOMMESSA" );
    interfaccia. scrivi ( "SCOMMETTI SU UN ASTRONAVE E PROVA A
TRIPPLICARE LA TUA PUNTATA!!!" );
    storia_gioco. insStoria ( stringa_comando , "hai guardato il simulatore di corse di
astronavi" );
}

```

Infine per far sì che le modifiche vadano a buon fine, sono state apportate le seguenti aggiunte in astro.cpp

```

case 80 :
    azione_80 ();
    break ;

case 81 :
    azione_81 ();
    break -,

```

5.7.2 Strutture dati aggiuntive

L'integrazione dei due progetti di Disabato e Prò prevede anche la struttura dati intercambiabile aggiuntiva della coda dinamica con puntatori.

```

#ifndef CODAP_H_
#define CODAP_H_

```

//Realizzazione coda dinamica con puntatori di Prò Giuseppe

```

#include "Cella.h"
#include <iostream>
#include <cstdlib>

```

```

using namespace std;

template<class Cell> class Coda
{
public:

    //definizioni di tipo
    typedef Cell tipoelem;
    typedef Cella<Cell>* posizione; // puntatore di tipo cella

    //definizione costruttori e distruttori
    Coda();
    Coda(Coda&);
    ~Coda();

    //specifica degli operatori
    void creacoda();
    bool codavuota() const;
    tipoelem leggicoda() const;
    void incoda(tipoelem);
    void fuoricoda();
    void svuota();

private :
    posizione testa; //puntatore alla testa
    posizione coda; //puntatore alla coda
};

```

```
template<class Cell> Coda<Cell>::Coda()
```

```
{
```

```
    creacoda();
```

```
}
```

```
template<class Cell> Coda<Cell>::Coda(Coda<Cell>& codaorig)
```

```
{
```

```
    creacoda();
```

```
    tipoelem temp;
```

```
    Coda<Cell> comodo; //creazione coda d'appoggio
```

```
    while (!codaorig.codavuota())// fintanto che la coda di partenza non è vuota esegui:
```

```
{
```

```
    comodo.incoda(codaorig.leggicoda()); //copio l'elemento della testa della coda  
    originaria in quella d'appoggio
```

```
    codaorig.fuoricoda(); //distruggo l'elemento in testa alla coda originaria
```

```
}
```

```
    while (!comodo.codavuota())// fintanto che la coda di appoggio non è vuota  
    esegui:
```

```
{
```

```
    temp=comodo.leggicoda();
```

```
    comodo.fuoricoda();
```

```
    incoda(temp); //copia nella nuova coda
```

```
    codaorig.incoda(temp); //ripristino coda originaria
```

```
}
```

```
}
```

```
template<class Cell> Coda<Cell>::~Coda() //elimina la coda
```

```
{
```

```
    while (!codavuota()) // se la coda non è vuota
```

```
{
```

```

        fuoricoda(); //elimino ogni singolo elemento
    }

delete coda; //cancellazione puntatore testa
delete testa; //cancellazione puntatore coda
}

```

```

template<class Cell> void Coda<Cell>::creacoda() //crea coda vuota
{
    testa=NULL; //inizializzazione testa
    coda=NULL; //inizializzazione coda
}

```

```

template<class Cell> bool Coda<Cell>::codavuota() const //restituisce true se la
coda è vuota, false altrimenti
{
    return ((testa==NULL) && (coda==NULL));
}

```

```

template<class Cell> Cell Coda<Cell>::leggicoda() const //legge l'elemento in
testa
{
    if (!codavuota()) //precondizione coda non vuota
        return (testa->leggicella());
}

```

```

template<class Cell> void Coda<Cell>::incoda(tipoelem elemento) //aggiunge un
elemento in coda
{
    Cella<Cell>* temp=new Cella<Cell>; //creazione nuovo elemento
    temp->scrivicella(elemento); //scrittura valore
}

```

```

temp->scrivisucc(NULL); //Mettiamo NULL , poichè il successivo non c'è
if (!codavuota()) {coda->scrivisucc(temp);} // se la coda non era vuota allora
l'elemento che era in coda ha come successivo sarà il temp
else {testa=temp;} //altrimenti è in testa
coda=temp; //in ogni caso l'elemento appena inserito andrà in coda
}

```

```

template<class Cell> void Coda<Cell>::fuoricoda() //estrae l'elemento in testa
{
if (!codavuota()) //se la coda non è vuota
{
    bool uguale=false; //indica se testa e coda sono uguali
    if (testa==coda) {uguale=true;} // se testa e coda puntano allo stesso elemento
    posizione temp=testa; //creo un puntatore che punta l'elemento da eliminare
    (quello in testa)
    testa=testa->leggisucc();// testa diventa il successivo di se stesso
    delete (temp); //elimino l'elemento precedentemente in testa
    if (uguale) { coda=NULL; } //se coda era uguale a testa (quindi un solo
    elemento) adesso la coda è vuota quindi coda=NULL
}
}

//operatore ausiliare

```

```

template<class Cell> void Coda<Cell>::svuota() //svuota la coda
{

```

```

while (!codavuota())
{

```

```

        fuoricoda();
    }
}

template<class Cell> ostream& operator<<(ostream& os, Coda<Cell>& codaorig
//sovraffare output

{
    Coda<Cell> comodo;
    while (!codaorig.codavuota())
    {
        os<<codaorig.leggicoda();
        comodo.incoda(codaorig.leggicoda());
        codaorig.fuoricoda();
        if (!codaorig.codavuota()) os<<",";
    }
    while (!comodo.codavuota()) //ripristino la coda
    {
        codaorig.incoda(comodo.leggicoda());
        comodo.fuoricoda();
    }
    return(os);
}

#endif

```

5.8 AUDITORIUM

5.8.1 Gioco.h

Inserite le dichiarazioni delle variabili, delle strutture dati necessarie per il funzionamento del Jukebox e del pannello delle luci e le funzioni di salvataggio e ripristino del Jukebox e delle luci.

```
Void salvaJukeBoxLuci(); //salva lo stato del juke e delle luci  
void caricaJukeBoxLuci() //ripristina lo stato del juke e delle luci
```

```
Canzone canzone; Inizializzazione classe canzone  
Lista<Canzone> canzoni; Inizializzazione della lista di  
canzoni del Jukebox  
Lista<Canzone>::posizione canzoneScelta;  
bool jukeboxAttivo; //Verifica se il jukebox è usato  
int canzoneInRiproduzione; //Serve per il salvataggio  
Coda<Canzone> playlist; //Salva la playlist dei brani ascoltati  
  
Luce luce;  
Insieme<Luce> luciAuditoriumAccese; - permette di vedere  
quali sono le luci accese nell'auditorium  
string nome_luci_auditorium[6] = {"fittizio", "sulla destra,  
avanti", "sulla sinistra, avanti", "sulla destra, dietro", "sulla  
sinistra, dietro", "sul palco"}; - array di stringhe dei nomi delle luci
```

5.8.2 Astro.h

Inseriti i seguenti metodi per inizializzare il Jukebox e per rendere interagibili gli oggetti presenti nell'auditorium:

```
void inizializza_Jukebox(); //permette di inizializzare il jukebox.  
separato per ordinare il codice  
  
void azione_105(); //usa proiettore rotto  
void azione_106(); //guarda schermo auditorium  
void azione_107(); //guarda dalla finestra dell'auditorium  
void azione_108(); //Usa il microfono spento nell'auditorium  
void azione_109(); //Usa il Jukebox nell'auditorium  
void azione_110(); //Guarda gli strumenti musicali  
void azione_111(); //Suona uno strumento musicale  
void azione_112(); //Interagisci col pannello delle luci  
void azione_113(); //Interagisci con la scacchiera
```

5.8.3 Astro.cpp

Inserimento delle parole chiave nel vocabolario del gioco:

```
vocabolario.inserisci("Auditorium", 16);  
vocabolario.inserisci("proiettore rotto", 80);
```

```
vocabolario.inserisci("schermo", 60);
vocabolario.inserisci("finestra", 90);
vocabolario.inserisci("microfono", 90);
vocabolario.inserisci("strumenti", 88);
vocabolario.inserisci("strumento", 88);
vocabolario.inserisci("suona", 72);
vocabolario.inserisci("jukebox", 56);
vocabolario.inserisci("pannello", 70);
vocabolario.inserisci("scacchiera", 73);
```

Inserimento delle azioni per interagire con gli oggetti dell'auditorium:

```
azioni.inserisci(328680, 105); //Accendi il proiettore rotto
azioni.inserisci(321060, 106); //Guarda lo schermo
azioni.inserisci(321090, 107); //Guarda dalla finestra
azioni.inserisci(323990, 108); //Parla al microfono
azioni.inserisci(322956, 109); //Interagisci con il Jukebox
azioni.inserisci(321088, 110); //Guarda gli strumenti musicali
azioni.inserisci(327288, 111); //Usa gli strumenti musicali
azioni.inserisci(322970, 112); //Usa il pannello delle luci
azioni.inserisci(321073, 113); //Guarda la scacchiera
```

Inserimento degli oggetti nell'Auditorium

```
oggetti.inserisci(Oggetto("un proiettore rotto", 80, -32));
oggetti.inserisci(Oggetto("uno schermo", 60, -32));
oggetti.inserisci(Oggetto("una finestra grande", 90, -32));
oggetti.inserisci(Oggetto("un microfono", 90, -32));
oggetti.inserisci(Oggetto("dei strumenti musicali", 88, -32));
oggetti.inserisci(Oggetto("una scacchiera sulla cattedra", 73, -32));
oggetti.inserisci(Oggetto("un jukebox antico", 56, -32));
oggetti.inserisci(Oggetto("un pannello di controllo delle luci", 70, -32));
```

```
Inizializza_Jukebox(){ //Si potrebbe creare un sistema di caricamento da file
    jukeboxAttivo = false; // Questo flag è usato per far interagire il giocatore col jukebox
    fin quando vorrà
    canzoni.svuota();
    canzoneScelta = canzoni.primolista(); // prima posizione della lista

    // INIZIALIZZAZIONE DELLE CANZONI NEL JUKEBOX
    canzone.set_nome("Yellow Submarine");
    canzone.set_artista("Beatles");
    canzone.set_anno(1966);
    canzoni.inslista(canzone, canzoneScelta);

    canzone.set_nome("We will rock you");
    canzone.set_artista("Queen");
    canzone.set_anno(1977);
    canzoni.inslista(canzone, canzoneScelta);

    canzone.set_nome("The man who sold the World");
    canzone.set_artista("David Bowie");
    canzone.set_anno(1970);
    canzoni.inslista(canzone, canzoneScelta);

    canzone.set_nome("Hotel California");
    canzone.set_artista("The Eagles");
    canzone.set_anno(1976);
    canzoni.inslista(canzone, canzoneScelta);

    canzone.set_nome("Crazy");
    canzone.set_artista("Willie Nelson");
    canzone.set_anno(1961);
    canzoni.inslista(canzone, canzoneScelta);
```

```

canzone.set_nome("Wish you were here");
canzone.set_artista("Pink Floyd");
canzone.set_anno(1975);
canzoni.inslista(canzone, canzoneScelta);

canzone.set_nome("Blowin' in the wind");
canzone.set_artista("Bob Dylan");
canzone.set_anno(1963);
canzoni.inslista(canzone, canzoneScelta);

canzone.set_nome("Imagine");
canzone.set_artista("John Lennon");
canzone.set_anno(1971);
canzoni.inslista(canzone, canzoneScelta);

canzone.set_nome("Generale");
canzone.set_artista("Francesco De Gregori");
canzone.set_anno(1978);
canzoni.inslista(canzone, canzoneScelta);

canzone.set_nome("Un giudice");
canzone.set_artista("Fabrizio De Andre'");
canzone.set_anno(1971);
canzoni.inslista(canzone, canzoneScelta);

canzoneScelta =canzoni.primolista(); //Verra' usata questa variabile per tenere
//conto di quale canzone sta suonando il jukebox
canzonelnRiproduzione = 0;
}

```

Implementazione delle azioni che permettono al giocatore di interagire con gli oggetti dell'Auditorium

5.8.3.1 *Accendi il proiettore rotto*

```
void Astro::azione_105(){  
    interfaccia.scrivi("e' un proiettore non funzionante, la lente e' rottata");  
}
```

5.8.3.2 *Guarda lo schermo nell'auditorium*

```
void Astro::azione_106(){  
    interfaccia.scrivi("Lo schermo non mostra nulla poiche' il proiettore non  
    e' acceso.");  
}
```

5.8.3.3 *Guarda la finestra chiusa*

```
void Astro::azione_107(){  
    interfaccia.scrivi("Lo spettacolo deve essere mozzafiato... peccato che il  
    sistema di sicurezza abbia attivato la chiusura di tutte le finestre della  
    Neutronia...");  
}
```

5.8.3.4 *Parla al microfono*

```
void Astro::azione_108(){  
    interfaccia.scrivi("La mia voce riecheggia in tutto l'Auditorium! Meglio  
    smetterla prima di ricevere una lavata di capo...");  
}
```

5.8.3.5 *Interagisci con il Jukebox Antico*

```
void Astro::azione_109(){  
    string decisione; //per contenere la decisione dell'utente  
    interfaccia.scrivi("È un antico Jukebox! L'acustica qua e' perfetta!");  
    jukeboxAttivo = true;  
    while(jukeboxAttivo){  
        canzone = canzoni.leggilista(canzoneScelta);  
        playlist.incoda(canzone); //inserisce nella coda la canzone in  
        riproduzione.  
        cout << "\n";
```

```

cout << canzone; //possibile grazie all'overload dell'operatore
interfaccia.scrivi("Posso andare alla prima canzone, a quella
precedente o a quella successiva. Cosa faccio?");

cin >> decisione;

if ( (decisione.compare("prima") == 0) ||
(decisione.compare("Prima") == 0) ||
(decisione.compare("PRIMA") == 0) ){
    canzoneScelta = canzoni.primolista();
    canzoneInRiproduzione = 0
}

else if( (decisione.compare("successiva") == 0) ||
(decisione.compare("Successiva") == 0) ||
(decisione.compare("SUCCESSIVA") == 0) ){

    if ( canzoni.finlista(canzoni.succlista(canzoneScelta)) ){

        interfaccia.scrivi("Poiche' e' l'ultima canzone, il Jukebox e'
tornato alla prima canzone.");
        canzoneScelta = canzoni.primolista();
        canzoneInRiproduzione = 0
    }

    else {

        canzoneScelta = canzoni.succlista(canzoneScelta);
        canzoneInRiproduzione++;
    }

}

else if ( (decisione.compare("precedente") == 0) ||
(decisione.compare("Precedente") == 0) ||
(decisione.compare("PRECEDENTE") == 0) ){

    if (canzoneScelta == canzoni.primolista() ){

        canzoneScelta = canzoni.primolista();
        interfaccia.scrivi("Questo jukebox non puo' tornare dalla prima
canzone all'ultima!");
        canzoneInRiproduzione = 0;
    }

    else {

        canzoneScelta = canzoni.prelista(canzoneScelta); }

        canzoneInRiproduzione -= -;
    }

    else {

        jukeboxAttivo = false;
    }

}

```

```

cout << "\n" << "\n" << "\n";
interfaccia.scrivi("Meglio che vada a salvare la Neutronia..."); 
interfaccia.scrivi("Ho ascoltato, nel seguente ordine, queste canzoni: ");
cout << "\n";
int cnt = 0; //Per far stampare il numero affianco alla canzone
//Stampa tutta la playlist.
while(!playlist.codavuota()){
    cnt++;
    canzone = playlist.leggicoda();
    playlist.fuoricoda();
    cout << cnt << " ) " << canzone.get_nome() << "\n"; //Evito di stampare
    tutte le info del brano con il COUT overloaded }      }

```

5.8.3.6 *Guarda gli strumenti Musicali*

```

void Astro::azione_110(){

interfaccia.scrivi("Sul palco dell'Auditorium sono presenti una chitarra, un
violino e un pianoforte"); }

```

5.8.3.7 *Suona uno strumento musicale*

L'IF permette di scegliere quale strumento suonare

```

void Astro::azione_111(){

string strumentoDaSuonare;

interfaccia.scrivi("Potrei provare a suonare il violino, il pianoforte o la chitarra... ");
interfaccia.scrivi("Quale scelgo?");

cin >> strumentoDaSuonare;

if (strumentoDaSuonare.compare("violino") ==0){

    interfaccia.scrivi("Provo a sfregare l'archetto sulle corde del violino... ma
i risultati sono disastrosi..."); }

else if (strumentoDaSuonare.compare("pianoforte") ==0){

    interfaccia.scrivi("Nell'auditorium rieccheggiano le poche note
che conosco della Sinfonia numero 9."); }

else if (strumentoDaSuonare.compare("chitarra") ==0){

    interfaccia.scrivi("Forse in un'altra vita ero capace di suonarla...
ma non in questa."); }

else { interfaccia.scrivi("Forse e' meglio lasciar stare..."); }

```

```
}
```

5.8.3.8 Interagisci con le luci dell'auditorium

```
void Astro::azione_112(){

    string nome_luce;
    int bottonePremuto;
    bool inputValido = true;

    interfaccia.scrivi("È un pannello di controllo delle luci dell'Auditorium");
    interfaccia.scrivi("Vedo dei bottoni che si illuminano numerati, con un'etichetta affianco");
    cout << "\n";
    if (luciAuditoriumAccese.insiemevuoto() ){
        interfaccia.scrivi("Tutte le luci dell'Auditorium sono spente");}
    else {
        for(int i= 1; i <6; i++){
            luce.setNome(nome_luci_auditorium[i]);
            if (luciAuditoriumAccese.appartiene(luce)){
                cout << "La luce " << luce << " e' accesa \n"; } } }

    //Si potrebbero usare direttamente i nomi delle luci per scrivere le etichette affianco ai bottoni,
    cout << "\n";
    interfaccia.scrivi("1. Destra Avanti");
    interfaccia.scrivi("2. Sinistra Avanti");
    interfaccia.scrivi("3. Destra Dietro");
    interfaccia.scrivi("4. Sinistra Dietro");
    interfaccia.scrivi("5. Palco");
    cout << "\n";
    interfaccia.scrivi("Quale bottone premo?");
    cin >> bottonePremuto;
    if ( (cin.fail() ) || (bottonePremuto >5) || (bottonePremuto<1)){
        interfaccia.scrivi("Meglio che vada a salvare la Neutronia...");}
```

```

        inputValido = false;} else{
nome_luce=nome_luci_auditorium[bottonePremuto]; }

if (inputValido){
    luce.setNome(nome_luce);
    if (!luciAuditoriumAccese.appartiene(luce)){
        luciAuditoriumAccese.inserisci(luce);
        cout << "La luce che si trova " << luce << " e' ora accesa. \n" ;
    } else {
        luciAuditoriumAccese.cancella(luce);
        cout << "La luce che si trova " << luce << " e' ora spenta. \n" ;
    }
}
}

```

5.8.3.9 Guarda la scacchiera

```

void Astro::azione_113(){
interfaccia.scrivi("La scacchiera, per qualche motivo, ha 8 regine schierate.");
interfaccia.scrivi("Sono disposte in maniera tale che nessuna possa mangiare
l'altra.");
interfaccia.scrivi("Mi chiedo il motivo di questa curiosa disposizione...");}

```

Nella procedura esegui_specifiche(int a, Mappa &M) sono stati aggiunti i seguenti case:

//accendi proiettore Auditorium

```

case 105:
    azione_105(); break;

```

//Guarda lo schermo nell'Auditorium

```
case 106:  
    azione_106(); break;
```

//Guarda la finestra nell'Auditorium

```
case 107:  
    azione_107(); break;
```

//Parla col microfono dell'Auditorium

```
case 108:  
    azione_108(); break;
```

//Utilizza il Jukebox presente nell'auditorium

```
case 109:  
    azione_109(); break;
```

//Osserva gli strumenti musicali presenti nell'auditorium

```
case 110:  
    azione_110(); break;
```

//Suona uno strumento musicale

```
case 111:  
    azione_111(); break;
```

//Interagisci col pannello delle luci nell'auditorium

```
case 112:  
    azione_112(); break;
```

//Guarda scacchiera

```
case 113:  
    azione_113(); break;
```

5.8.4 Gioco.cpp

//Salva lo stato del Jukebox e delle luci. Per il jukebox usa un intero per tenere traccia della “”posizione”” a cui l'utente si è fermato, mentre per le luci salverà sul file tutte le luci che, nel momento del salvataggio, appartengono all'insieme delle luci accese

```

void Gioco::salvaJukeBoxLuci(){
    ofstream nuovofile("JukeLuci.txt");
    nuovofile << canzoneInRiproduzione << '\n';
    for (int i = 0;
        i < sizeof(nome_luci_auditorium)/sizeof(nome_luci_auditorium[0]);
        i++){
        luce.setNome(nome_luci_auditorium[i]);
        if (luciAuditoriumAccese.appartiene(luce)){
            nuovofile << '-' << nome_luci_auditorium[i] << '\n';
        }
    }
    nuovofile << '?';
    nuovofile.close();  }

```

Ripristina lo stato del jukebox e delle luci accese. Col primo numero scritto nel file si vede quante volte bisogna eseguire l'operatore succlista (partendo dal primo elemento) eventuali stringhe scritte indicano, invece, quali sono le luci che sono state accese, quindi verranno inserite immediatamente nell'insieme delle luci accese.

```

void Gioco::caricaJukeBoxLuci(){
    ifstream file("JukeLuci.txt");  string rigafile;
    getline(file,rigafile);
    canzoneScelta = canzoni.primolista();
    int puntoRipresaJuke = atoi(rigafile.c_str() );

    for (int i = 0; i< puntoRipresaJuke; i++){
        canzoneScelta = canzoni.succlista(canzoneScelta); }

    while(file.get()!='?'){
        getline(file,rigafile);
        cout << rigafile << "\n";
        luce.setNome(rigafile);
        if ( !luciAuditoriumAccese.appartiene(luce) ){
            luciAuditoriumAccese.inserisci(luce);
        }
    }
    file.close(); }

```

5.9 STAZIONE

Per inserire il nuovo luogo, i nuovi oggetti e le varie azioni sono stati modificati i seguenti file:

- Mappa.nav
- Descrizioni (33.txt)

- Astro.cpp
- Astro.h

5.9.1 Mappa.nav

• È stato incrementato il numero di stanze (da 32 a 33).

• È stato inserito il nuovo luogo “Stazione”:

33,Nella Stazione,001200000000,via 2,1,1

Dove:

- 33 è il codice del luogo
- Nella Stazione è il nome del luogo
- 001200000000 è il codice delle direzioni
- via 2,1,1 rappresenta la via

• È stata modificata la linea di codice relativa all’ufficio/deposito, aggiornando il codice delle direzioni ed aggiungendo una nuova via:

12,Nell'ufficio/deposito,332414000200,via 1,1,1,1,6,2,2,2,2,1,1

• Aggiungendo le seguenti righe di codice, è stata specificata la relazione che sussiste tra il luogo appena creato e l’ufficio/deposito:

12,33 via 1,nord,2,2,1,1

33,12 via 1,sud,2,2,1,1

Questa relazione è espressa in termini di:

LuogoPartenza – LuogoDestinazione – Via – Direzione – Lunghezza – Tempo – Asfalto – Pedaggio

5.9.2 Descrizioni (33.txt)

Ho aperto la cartella Descrizioni ed ho creato il file “33.txt” che conterrà le descrizioni relative al luogo “Stazione”:

di nuovo nella stazione

nella stazione

nuovamente nella stazione

ancora nella stazione

5.9.3 Astro.cpp

Nel file astro.cpp sono state apportate le seguenti modifiche:

5.9.3.1 Inserimento vocaboli

```
//INIZIO modifiche SALVATORE VESTITA
vocabolario.inserisci("siediti", 94);

//FINE modifiche SALVATORE VESTITA
```

5.9.3.2 Inserimento azioni

```
//INIZIO modifiche SALVATORE VESTITA
azioni.inserisci(339471, 114); // "Siediti sulla panchina"
azioni.inserisci(332980, 115); // "usa biglietteria"
```

```

azioni.inserisci(332578, 116); // "leggi giornale"
azioni.inserisci(330555, 117); // "sali sul treno"
//FINE modifiche SALVATORE VESTITA

```

```

//INIZIO modifiche SALVATORE VESTITA
vocabolario.inserisci("sulla", 7);
vocabolario.inserisci("sul", 7);
//FINE modifiche SALVATORE VESTITA

```

```

//INIZIO modifiche SALVATORE VESTITA
vocabolario.inserisci("biglietteria", 80);
vocabolario.inserisci("giornale", 78);
vocabolario.inserisci("treno", 55);
vocabolario.inserisci("panchina", 71);
//FINE modifiche SALVATORE VESTITA

```

5.9.3.3 Inserimento oggetti

```

//INIZIO modifiche SALVATORE VESTITA
oggetti.inserisci(Oggetto("una biglietteria", 80, -33));
oggetti.inserisci(Oggetto("un frammento di giornale", 78, 33));
oggetti.inserisci(Oggetto("un treno", 55, -33));
oggetti.inserisci(Oggetto("una panchina", 71, -33));
//FINE modifiche SALVATORE VESTITA

```

5.9.3.4 Implementazione azioni

L'azione 114 (sedersi sulla panchina) è stata pensata col fine di far perdere del tempo al giocatore, infatti questo potrà semplicemente sedersi ed alzarsi dalla panchina. Nel momento in cui il giocatore si siede, gli verrà sottratto 1 punto al tempo. Infatti il giocatore vedrà il messaggio "Non sprecare il tuo tempo restando seduto!!". L'utente una volta seduto potrà solamente alzarsi. Una volta alzato potrà compiere normalmente tutte le altre azioni.

```

void Astro::azione_114() {
    interfaccia.scrivi("Ti sei seduto.");
    storia_gioco.insStoria(stringa_comando, "Hai deciso di sederti");
    //Aggiorna storia gioco
    cout << "Ecco il tuo tempo a disposizione: " << tempo;
}

```

```

interfaccia.a_capo();

tempo--;

interfaccia.scrivi("Non sprecare il tuo tempo restando seduto!!!!");
interfaccia.a_capo();

string risp;

interfaccia.scrivi("Vuoi alzarti? ");
cin >> risp;

while(risp!="si" && risp!="SI" && risp!="Si" && risp!="sI") {
    interfaccia.scrivi("E ora ti vuoi alzare? ");
    cin >> risp;
}

cout << "Ti sei alzato";
storia_gioco.insStoria(stringa_comando , "Hai deciso di alzarti");
//Aggiorna storia gioco
}

```

L'azione 115 (usare la biglietteria) consente al giocatore di acquistare biglietti per il treno. Nel progetto (da integrare) di Semeraro il tutto veniva gestito tramite una variabile di tipo string “selezione” e, tramite alcune semplici istruzioni di controllo (if-else), verificava se ciò che inseriva in input l’utente (selezione) era uguale o meno alle destinazioni disponibili e quindi procedeva in tale direzione.

La novità presente nel mio progetto è l’utilizzo della struttura dati Dizionario per gestire i biglietti. Ho creato quindi un Dizionario chiamato “Biglietti”, formato dalla coppia <chiave, valore>, dove la chiave è il codice (univoco) del luogo (un intero), il valore è rappresentato dalla classe Biglietto. Ho creato la classe Biglietto, in cui il biglietto è formato dalla coppia nome della destinazione (stringa) e costo (intero).

Ho scelto di utilizzare il Dizionario in quanto l’inserimento viene effettuato solo quattro volte (per inserire i biglietti relativi a “Aula”, “Ristorante”, “Stadio”, “Banca”). Invece la ricerca viene effettuata spesso! Come ben sappiamo, nel dizionario la massima esigenza è proprio quella di ritrovare in modo estremamente efficiente un qualunque elemento di interesse.

Un’altra novità presente nel mio progetto è l’utilizzo della struttura dati Coda, per simulare le persone in coda alla biglietteria. Il tutto viene effettuato in maniera casuale tramite la variabile “casuale” che assumerà un valore intero da 0 a 9.

```

bool biglietto=false;

void Astro::azione_115() {
    Biglietto b1("aula", 100);

```

```

Biglietto b2("ristorante", 80);
Biglietto b3("stadio", 50);
Biglietto b4("banca", 49);
Dizionario<int, Biglietto> Biglietti;
Biglietti.inserisci(25,b1);
Biglietti.inserisci(21,b2);
Biglietti.inserisci(17,b3);
Biglietti.inserisci(15,b4);

//CREO UNA CODA PER SIMULARE IL NUMERO DI PERSONE IN CODA ALLA BIGLIETTERIA
Coda<int> c;
int casuale, numpersincoda, i;
casuale=rand()%10;
numpersincoda=0;
i=0;

while(i<=casuale) {
    i++;
    c.incoda(i);
}
numpersincoda=i; //variabile contenente il numero di persone in coda

string input;//variabile che conterrà l'input dato dall'utente
float denaro;//variabile che conterrà il denaro disponibile

if(portafoglio.hai_Portafoglio(oggetti)) {
    while(input!="si" && input!="s" && input!="Si" && input!="SI" && input!="no" && input!="NO" && input!="No" && input!="n" && input!="N") {
        if(numpersincoda>1) {
            interfaccia.scrivi_parziale(numpersincoda);
            interfaccia.scrivi(" persone sono in coda ad attendere il proprio turno, desidera attendere? [si/no] ");
        }
        else if(numpersincoda==0) {
            interfaccia.scrivi("È il tuo turno vuoi prenotare un biglietto? [si/no] ");
        }
    }
}

```

```

    else if (numpersincoda==1) {

        interfaccia.scrivi("C'e' una sola persona in coda,
desidera attendere? [si/no]");

    }

    cin>>input;

    if (input=="si" || input=="s" || input=="S" ||
input=="Si" || input=="SI" || input=="sI") {

        tempo-=numpersincoda;

        i=0;

        while (i<=casuale) {

            i++;

            c.fuoricoda();

        }

        cout << "\n\n\n";
interfaccia.scrivi("#####"); //messaggio di output per la biglietteria
    }

    interfaccia.scrivi("####" BENVENUTO IN FERROVIE PER LO SPAZIO
####);

    interfaccia.scrivi("####" DOVE DESIDERÀ ANDARE?
####);

    interfaccia.scrivi("#### SCEGLIERE TRA LE SEGUENTI DESTINAZIONI:
####");

    interfaccia.scrivi("#### -25: AULA (costo=100)
####");

    interfaccia.scrivi("#### -21: RISTORANTE (costo=80)
####");

    interfaccia.scrivi("#### -17: STADIO (costo=55)
####");

    interfaccia.scrivi("#### -15: BANCA (costo=49)
####");

    interfaccia.scrivi("#### -0: (PER USCIRE)
####");

interfaccia.scrivi("#####");
interfaccia.scrivi("#####");

    interfaccia.scrivi("#### N.B I BIGLIETTI ACQUISTATI NON SONO
####");

    interfaccia.scrivi("#### RIMBORSABILI PER ALCUN MOTIVO
####");
interfaccia.scrivi("#####");
interfaccia.a_capo();

```

```

int codiceinserito;

int costo;

interfaccia.a_capo();

cout << "INSERISCI CODICE LUOGO: ";

cin >> codiceinserito;

if(codiceinserito==0) {
    interfaccia.scrivi("### HAI DECISO DI ANNULLARE L'OPERAZIONE!
###");
}

while(!Biglietti.appartiene(codiceinserito) && codiceinserito!=0) {
    interfaccia.scrivi("#### DESTINAZIONE NON VALIDA!
####");
    cout << "INSERISCI NUOVAMENTE IL CODICE LUOGO: ";
    cin >> codiceinserito;
}

if(Biglietti.appartiene(codiceinserito)){
    Biglietto b;
    b = Biglietti.recupera(codiceinserito);
    costo=b.getCosto();
    string destinazione=b.getDestinazione();
    std::transform(destinazione.begin(), destinazione.end(),
destinazione.begin(), ::toupper);

    if(portafoglio.get_contanti()>=costo) {
        cout <<"\n\nHAI ACQUISTATO UN BIGLIETTO PER: " << destinazione
<< "!";
        denaro=portafoglio.get_contanti();
        biglietto=true;
        denaro-=costo;
        portafoglio.set_contanti(denaro); //aggiornamento denaro
    }
    else{
        interfaccia.a_capo();
        interfaccia.scrivi("ATTENZIONE! Non hai abbastanza soldi.");
    }
}

```


L'azione 116 (leggere il giornale) dovrebbe consentire al giocatore di leggere il giornale, in realtà questo risulta frammentato.

```
void Astro::azione_116() {
    interfaccia.scrivi("#### Il giornale e' troppo frammentato, non riesco
a leggere ####");
}
```

L'azione 117 (salire sul treno) consente al giocatore di salire sul treno, ma questo risulta guasto! Quindi il giocatore avrà perso inutilmente il proprio tempo ed il proprio denaro.

```
void Astro::azione_117() {
    if(bigietto==true) {
        interfaccia.scrivi("Din, don, siamo spiacenti di informare la clientela che a
seguito di un guasto il treno per");
        interfaccia.scrivi("Aula, Ristorante, Stadio, Banca, non potra' partire, ci
scusiamo per il disagio, grazie e arrivederci.");
        interfaccia.a_capo();
    } else if(bigietto==false) {
        interfaccia.scrivi("Per salire sul treno e' necessario acquistare un
bigietto");
        interfaccia.a_capo();
    };
} //FINE modifiche SALVATORE VESTITA
```

5.9.4 Astro.h

```
//INIZIO modifiche SALVATORE VESTITA
void azione_114();//Siediti sulla panchina
void azione_115();//Usa bigietteria
void azione_116();//leggi giornale
void azione_117();//Sali sul treno
//FINE modifiche SALVATORE VESTITA
```

5.9.5 Bigietto.h

```
#ifndef BIGLIETTO_H
#define BIGLIETTO_H
#include <string>
using namespace std;
```

```

class Biglietto {
public:
    Biglietto(string, int);
    Biglietto();
    ~Biglietto();
    string getDestinazione() const;
    int getCosto() const;
private:
    string destinazione;
    int costo;
};

#endif

```

5.9.6 Biglietto.cpp

```

#include "Biglietto.h"

using namespace std;

Biglietto::Biglietto(string d, int c) {
    destinazione=d;
    costo=c;
}

Biglietto::Biglietto() {
    destinazione="";
    costo=0;
}

Biglietto::~Biglietto() {
}

string Biglietto::getDestinazione() const{
    return destinazione;
}

int Biglietto::getCosto() const{

```

```
    return costo;  
}
```

5.10 PALESTRA

Di seguito saranno riportate tutte le modifica apportate al progetto (base) di Vestita per realizzare le funzionalità mancanti prese dal progetto (da integrare) di Savino.

Per inserire il nuovo luogo, i nuovi oggetti e le varie azioni sono stati modificati i seguenti file

- Mappa.nav
- Descrizioni (34.txt)
- Astro.cpp
- Astro.h
- Gioco.cpp

Sono stati inoltre importati i seguenti file

- Statofisico.h
- Statofisico.cpp
- Palestra.h
- Palestra.cpp
- Scheda.h
- Scheda.cpp

5.10.1 Mappa.nav

- È stato incrementato il numero di stanze (da 33 a 34)
- È stato inserito il nuovo luogo “Palestra”:

34,Nella Palestra,000000000500,via 1,1,1

Dove:

- 34 è il codice del luogo
- “Nella Palestra” è il nome del luogo
- 000000000500 è il codice delle direzioni
- Via 2,1,1 rappresenta la via

5,Nella cabina del secondo pilota,001802171634,via 2,2,2,via 7,2,2

- È stata modificata la linea di codice relativa alla cabina del secondo pilota, che permette ora, scendendo di raggiungere la palestra

5.10.2 Descrizioni (34.txt)

È stato creato il file “34.txt” che conterrà le descrizioni relative al luogo “Palestra”:

di nuovo nella palestra
in una palestra
nuovamente nella palestra
ancora una volta nella palestra

5.10.3 Astro.cpp

Nel fine Astro.cpp sono state apportate le seguenti modifiche:

5.10.3.1 *Inserimento vocaboli*

```
//INIZIO modifiche VINCENZO GIANNUZZO
vocabolario.inserisci("panca", 57);
vocabolario.inserisci("tapis", 47);
vocabolario.inserisci("roulant", 47);
vocabolario.inserisci("scheda", 84);
vocabolario.inserisci("schede", 84);
//FINE modifiche VINCENZO GIANNUZZO
```

5.10.3.2 *Inserimento azioni*

```
//INIZIO modifiche VINCENZO GIANNUZZO
azioni.inserisci(342957, 118); //usa panca
azioni.inserisci(342947, 119); //usa tapis
azioni.inserisci(342999, 120); //usa terminale
azioni.inserisci(341099, 120); //guarda terminale
azioni.inserisci(342984, 121); //usa schede
//FINE modifiche VINCENZO GIANNUZZO
```

5.10.3.3 *Inserimento oggetti*

```
//INIZIO modifiche VINCENZO GIANNUZZO
oggetti.inserisci(Oggetto("una panca per gli addominali", 57, -34));
oggetti.inserisci(Oggetto("un tapis roulant", 47, -34));
oggetti.inserisci(Oggetto("un terminale informativo", 99, -34));
oggetti.inserisci(Oggetto("delle schede di allenamento", 84, -34));
//FINE modifiche VINCENZO GIANNUZZO
```

5.10.3.4 *Implementazione azioni*

L'azione 118 (usare la panca) chiede in input il nome delle ripetizioni desiderate, e successivamente chiama la funzione usaPanca della classe Palestre, passando il valore numerico delle ripetizioni, e lo stato del fisico "sFisico", che fa parte della classe "StatoFisico". Questa funzione modifica di "costituzione" e diminuirà il tempo rimanente in base alle ripetizioni effettuate.

```
void Astro::azione_118() //usa panca
{
    interfaccia.scrivi("Quante ripetizioni vuoi fare ?");
    int rip;
    string risp;
    getline(cin, risp);
    stringstream(risp) >> rip;
    Palestre::usaPanca(sFisico, rip);
}
```

L'azione 119 differisce di poco all'azione 118, ma questa volta useremo il tapis roulant, dove verrà chiesto all'utente, il numero di minuti che vuole trascorrere sul tapis roulant. Questa funzione modificherà lo stato fisico del giocatore "resistenza", e il tempo trascorso sul tapis roulant influenzare anche il tempo totale rimanente per finire il gioco.

```

void Astro::azione_119() // usa tapis

{
    interfaccia.scrivi("Per quanti minuti vuoi correre ?");

    int rip;
    string risp;

    getline(cin, risp);

    stringstream(risp) >> rip;

    Palestra::usaTapis(sFisico, rip);
}

```

L'azione 120 invoca la funzione di usare il terminale, contenuta sempre all'interno della classe "Palestra".

```

void Astro::azione_120() //usa/guarda terminale

{
    Palestra::usaTerminale();
}

```

L'azione 121, chiama la funzione di accedere alle schede, inglobata all'interno della classe "Palestra".

```

void Astro::azione_121()

{
    Palestra::usaSchede(sFisico);
}

```

5.10.4 Astro.h

```

//INIZIO modifiche VINCENZO GIANNUZZO

void azione_118(); //Usa la panca

void azione_119(); //Usa tapis

void azione_120(); //Leggi terminale informativo

void azione_121(); //Usa schede

//FINE modifiche VINCENZO GIANNUZZO

```

5.10.5 Gioco.cpp

Sono state inserite alcune modifiche alle procedure aggiorna_tempo(), Le modifiche erano necessarie affinché i valori di costituzione e resistenza di StatoFisico potessero influenzare rispettivamente il tempo perso e le probabilità essere feriti.

```
float tempo_perso = 0; //Modifica GIANNUZZO  
if (sazio <= 120 && sazio >= 80)  
{  
tempo_perso+= 1; //Modifica GIANNUZZO  
if (sazio <= 90 && sazio >= 80)  
interfaccia.scrivi("\nInizio a sentire un leggero languo-rino...");  
}  
else if (sazio <= 79 && sazio >= 50)  
{  
tempo_perso+= 2; //Modifica GIANNUZZO  
if (sazio <= 60 && sazio >= 50)  
interfaccia.scrivi("\nE' tutto il giorno che non mangio, do-vrei fare una pausa...");  
}
```

Negli altri casi “tempo_perso” non viene aggiornata perché la sazietà è troppo bassa e la penalità viene sottratta direttamente da “tempo”.

```
if (Salute.GetStatoSalute() <= 40)  
{  
tempo -= Salute.GetPenalita();  
}  
else //La penalità non viene ridotta se la salute è troppo bassa  
{  
tempo_perso += Salute.GetPenalita();  
}
```

Nella funzione “esegui”, la probabilità di ricevere ferite viene ridotta dalla resistenza del personaggio

```
if(!IsFisico.feritaEvitata(rand()))//Modifica GIANNUZZO  
{  
    Aggiorna_Ferite(); //Generatore Ferite  
}
```

5.10.6 Gioco.h

Viene aggiunta l'inclusione alla libreria "Palestra.h" e l'aggiunta di una variabile di tipo stato fisico alla riga 188.

```
StatoFisico sFisico;
```

5.10.7 StatoFisico.h

Classe che implementa le caratteristiche fisiche del personaggio

```
class StatoFisico  
{  
public:  
    StatoFisico();  
    ~StatoFisico();  
  
    inline float getCostituzione() { return costituzione; }  
    inline void setCostituzione(float cos) { costituzione = cos; }  
    inline float getResistenza() { return resistenza; }  
    inline void setResistenza(float res) { resistenza = res; }  
    bool feritaEvitata(int random);  
    float riduciTempo(float time);  
    const static int MAX_COS;  
    const static int MAX_RES;  
  
private:  
    float costituzione;  
    float resistenza;
```

5.10.8 StatoFisico.cpp

```
const int StatoFisico::MAX_COS = 50;
const int StatoFisico::MAX_RES = 50;
StatoFisico::StatoFisico()
{
    costituzione = 0;
    resistenza = 0;}
StatoFisico::~StatoFisico()
{
}
float StatoFisico::riduciTempo(float time)
{
    if (resistenza == MAX_RES && costituzione == MAX_COS)
    {
        time -= (time / 100) * 75;
    }
    else
    {
        time -= (time / 100) * costituzione;
    }
    return time;
}
bool StatoFisico::feritaEvitata(int random)
{
    bool fer = false;
    random = (random % 99) + 1;
    if (resistenza == MAX_RES && costituzione == MAX_COS)
    {
        fer = (random <= 75);
    }
    else
    {
        fer = (random <= resistenza);
    }
    return fer;
}
```

5.10.9 Palestra.h

Classe che implementa le funzionalità del luogo “Palestra”.

Contiene i metodi statici necessari per utilizzare gli oggetti presenti nella palestra:

```
class Palestra
{
public:
    static void usaPanca(StatoFisico& stato, int ripetizioni);
    static void usaTapis(StatoFisico& stato, int ripetizioni);
    static void usaSchede(StatoFisico& stato);
    static void usaTerminale();
```

Per la realizzazione dell’insieme delle schede sono state utilizzate in combinazione un dizionario e una lista.

Il primo contiene le schede sotto forma di coppie chiave-valore dove la chiave è la stringa contenente il nome della scheda, il valore è il contenuto informativo della scheda stessa. La seconda contiene i soli nomi delle schede, ed è utilizzata per la stampa a schermo delle schede esistenti in mancanza di metodi per iterare sulle chiavi del dizionario.

```
private:
    static Lista<string> listaSchede;
    static Dizionario<string,Scheda> schede;
    static void stampaSchede();
    static void creaScheda();
    static void cancellaSchede();
    static void eseguiScheda(StatoFisico& stato);
```

Per i dettagli implementativi di Palestra, si rimanda al file Palestra.cpp

5.10.10 Scheda.h

Questa classe rappresenta le singole schede di allenamento. Ogni scheda consiste di un attributo “nome” di tipo stringa e uno chiamato “esercizi” il cui tipo è una Coda<Esercizio> dove <Esercizio> è un tipo struct pubblico definito in Scheda.h e formato dai campi string tipo (ovvero “panca” o “tapis”) e int ripetizioni.

L’utilizzo della coda per rappresentare la lista di esercizi è dovuto alla necessità di mantenere l’ordine in essi verranno eseguiti al momento dell’utilizzo della scheda; tuttavia, poiché il processo di lettura di una coda è

distruttivo, si è rivelato necessario implementare un costruttore di copia nel file Coda.h. In questo modo il metodo Scheda::getEsercizi() può restituire una copia dell’intera coda preservando l’originale.

Per altri dettagli riguardanti l’implementazione si rimanda al file Scheda.cpp.

```
class Scheda
{
public:
    Scheda();
    Scheda(string);
    ~Scheda();
    struct Esercizio
    {
        string tipo;
        int ripetizioni;
    };
    string getNome();
    void setNome(string);
    void addEsercizio(string tipo, int ripetizioni);
    Coda<Esercizio> getEsercizi();
private:
    string nome;
    Coda<Esercizio> esercizi;
```

5.11 SALA GIOCHI

5.11.1 Aggiornamento mappa

Per implementare il luogo richiesto sono stati apportati i seguenti cambiamenti:

- All’interno del file “Mappa.nav” è stato incrementato il contatore dei luoghi da 36 a 37.
- Per accedere al luogo creato è stato modificato il codice del luogo banca. Questo, per permettere al giocatore di muoversi ad est, attraverso via 6, entrando in “Sala Giochi”:
15,In una banca,230036140000,via 6,2,2
- È stato inserito il nuovo luogo:
36,Nella Sala Giochi,000000150000,via 6,2,2
- Sono state aggiunte le due seguenti righe che permettono il controllo sui movimenti dei giocatori:
36,15 via 6,ovest,2,2,1,1 (spostamento dal luogo 36 al luogo 15)
15,36 via 6,est,2,2,1,1 (spostamento dal luogo 15 al luogo 36);
- Modifica del file 36.txt in descrizioni, contenete il seguente testo:

nuovamente nella Sala Giochi
sei nella Sala Giochi
ancora una volta nella Sala Giochi
per l'ennesima volta nella Sala Giochi

5.11.2 Aggiornamento dei vocaboli

All'interno del file Astro.cpp sono stati aggiunti i seguenti vocaboli:

```
vocabolario.inserisci("slotmachine", 86);
vocabolario.inserisci("5", 90);
vocabolario.inserisci("10", 91);
vocabolario.inserisci("20", 92);
vocabolario.inserisci("50", 93);
vocabolario.inserisci("100", 94);
vocabolario.inserisci("euro", 95);
```

5.11.3 Aggiornamento azioni

All'interno di Astro.cpp è stata aggiunta la seguente azione permessa nel luogo Sala Giochi:

azioni.inserisci(3600290086, 125); usa slot machine

All'interno di Astro.h sono state aggiunte le seguenti azioni:

void azione_125();//gioca alla slot machine;

All'interno di Astro.cpp sono state aggiunte le seguenti azioni:

case 125:

 azione_125();

 break;

5.11.4 Aggiornamento oggetti

È stato, inoltre, riempito l'insieme Slot-machine:

```
oggetti.inserisci(Oggetto("una slotmachine", 86,-36));
oggetti.inserisci(Oggetto("5 euro",90,-99)); //Luogo -99 in quanto non visibili
oggetti.inserisci(Oggetto("10 euro",91,-99));
oggetti.inserisci(Oggetto("20 euro",92,-99));
oggetti.inserisci(Oggetto("50 euro",93,-99));
oggetti.inserisci(Oggetto("100 euro",94,-99));
```

```
for(int indiceInsieme = 1; indiceInsieme<=31; indiceInsieme++)
{
```

```
    slotmachine.inserisci(indiceInsieme);
}
```

5.11.5 Aziona gioca slot-machine

All'intendo di astro.cpp andremo a inserire l'azione per giocare alla slot-machine:

```
void Astro::azione_125(){
    string risposta;
    string continua;
    int numeroEstratto;
    bool sentinella = true;
    int indiceDisponibilita;
    bool disponibilita5 = false;
    bool disponibilita10 = false;
    bool disponibilita20 = false;
    bool disponibilita50 = false;
    bool disponibilita100 = false;
    string giocare;
    interfaccia.scrivi("Benvenuto alla slotmachine!");
    interfaccia.scrivi("Puoi vincere questi premi:");
    indiceDisponibilita = 1;
    while(indiceDisponibilita<=16 && !disponibilita5)
    {
        if(slotmachine.appartiene(indiceDisponibilita))
            disponibilita5 = true;
        indiceDisponibilita++;
    }
    indiceDisponibilita = 17;
    while(indiceDisponibilita<=24 && !disponibilita10)
    {
        if(slotmachine.appartiene(indiceDisponibilita))
            disponibilita10 = true;
        indiceDisponibilita++;
    }
    indiceDisponibilita = 25;
    while(indiceDisponibilita<=28 && !disponibilita20)
    {
        if(slotmachine.appartiene(indiceDisponibilita))
            disponibilita20 = true;
        indiceDisponibilita++;
    }
    indiceDisponibilita = 29;
    while(indiceDisponibilita<=30 && !disponibilita50)
    {
        if(slotmachine.appartiene(indiceDisponibilita))
            disponibilita50 = true;
        indiceDisponibilita++;
    }
}
```



```

slotmachine.cancella(numeroEstratto);
}
else if(numeroEstratto>=25 && numeroEstratto<=28 && disponibilita20)
{
interfaccia.scrivi("Congratulazioni! Hai vinto 20 euro!");
soldi=portafoglio.get_contanti()+20.00;
portafoglio.set_contanti(soldi);
oggetti.set_luogo(31+numeroEstratto,0);
slotmachine.cancella(numeroEstratto);
}
else if(numeroEstratto>=29 && numeroEstratto<=30 && disponibilita50)
{
interfaccia.scrivi("Congratulazioni! Hai vinto 50 euro!");
soldi=portafoglio.get_contanti()+50.00;
portafoglio.set_contanti(soldi);
oggetti.set_luogo(31+numeroEstratto,0);
slotmachine.cancella(numeroEstratto);
}
else if(numeroEstratto>=31 && numeroEstratto<=31 && disponibilita100)
{
interfaccia.scrivi("Congratulazioni! Hai vinto 100 euro!");
soldi=portafoglio.get_contanti()+100.00;
portafoglio.set_contanti(soldi);
oggetti.set_luogo(31+numeroEstratto,0);
slotmachine.cancella(numeroEstratto);
}
else //nel caso in cui il numero  buono ma non c' pi disponibilita'
{
interfaccia.scrivi("Non hai vinto.");
interfaccia.scrivi("Ritenta, sarai piu' fortunato!");
}
}
else
{
interfaccia.scrivi("");
interfaccia.scrivi("Non hai vinto.");
interfaccia.scrivi("Andra' meglio la prossima volta!");
}
interfaccia.scrivi("");
interfaccia.scrivi("Vuoi giocare ancora?");
interfaccia.scrivi("1. Si");
interfaccia.scrivi("2. No");
cin >> continua;
if (continua == "1" || continua == "Si" || continua == "si" || continua == "SI" || continua == "s" ||
continua == "S")
{
risposta="gioca";
}
else{

```

```

if (continua == "2" || continua == "No" || continua == "no" || continua == "NO" || continua == "n"
|| continua == "N")
{
    risposta="esci";
}
else{
    nonvalido=true;
    interfaccia.scrivi("Comando non valido!}");
}
else{
cout<<"Mi dispiace, non hai soldi sufficienti. Soldi = 0.";
sentinella=false;
}
}
else if(risposta == "esci" || risposta == "Esci" || risposta == "e" || risposta == "E" || risposta == "2")
{
interfaccia.scrivi("");
interfaccia.scrivi("Arrivederci!");
sentinella = false;
}
else
nonvalido=true;
interfaccia.scrivi("Comando non valido!");
}
}

```

Nel file gioco.h viene dichiarato l'insieme slot-machine (di tipo InsiemeBool) ed una variabile che servirà a gestirlo nel salvataggio e nel caricamento:

```
#include "Insieme.h"

Insieme<int> slotmachine;

int numEuro;
```

5.11.6 Metodo load() e save()

Tenendo conto del denaro presente nella slot-machine, il metodo load() del file gioco.cpp è stato modificato come di seguito:

```
int slot;

svuotaInsieme(slotmachine);

file >> numEuro;

for(int j=1; j<=numEuro; j++)


```

```

{
    file >> slot;
    slotmachine.inserisci(slot);
}

```

Stessa cosa per il metodo save():

```

for(int k=1; k<=31; k++)
    if(slotmachine.appartiene(k))
        numEuro++;

```

```
file << numEuro << '\n';
```

```

for(int j=1; j<=31; j++)
    if(slotmachine.appartiene(j))
        file << j << '\n';

```

Gianluca Scatigna

5.12 IMPLEMENTAZIONE BIBLIOTECA

5.12.1 Aggiornamento mappa

Si modifica il numero di luoghi della prima riga del file impostandolo a 38

Aggiunta luogo Biblioteca con codice 37 e codice 28 per il comando scendi in scuola.

```

1. //mappa.nav
2. 37, Nella biblioteca, 00000000028, via 1, 2, 2

```

Aggiunta luogo Vetrina con codice 38 e codice 28 per il comando scendi in scuola.

```

1. //mappa.nav
2. 38, Nella vetrina, 00000000028, via 1, 2, 2

```

Modifica del luogo Scuola inserendo il codice 37 (Biblioteca) per il comando sali.

```
1. //mappa.nav
```

```
2. 28, Nella Scuola, 300129313700, via 1, 1, 1
```

5.12.2 Aggiornamento dei vocaboli

Per l'utilizzo del luogo è necessario utilizzare dei vocaboli identificati da un codice univoco di riferimento OO.

```
1. //astro.cpp  
2. vocabolario.inserisci("chiudi", 93);  
3. vocabolario.inserisci("libro", 95);  
4. vocabolario.inserisci("astronave", 90);  
5. vocabolario.inserisci("vetrina", 19);  
6. vocabolario.inserisci("equipaggio", 91);
```

*Gli oggetti considerati come libri devono avere come OO un intero al più di 90 per garantire il funzionamento delle azioni.

5.12.3 Aggiornamento azioni

```
1. //astro.cpp  
2. azioni.inserisci(3700250070, 126); //nuova azione etichetta su biblioteca  
3. azioni.inserisci(3700220019, 127); //nuova azione di apertura su vetrina  
4. azioni.inserisci(3800930019, 128); //nuova azione di chiusura su vetrina  
5. azioni.inserisci(250090, 129); //leggi libro astronave  
6. azioni.inserisci(250091, 129); //leggi libro equipaggio
```

5.12.4 Aggiornamento oggetti

LL 37 = Biblioteca

LL 38 = Vetrina

```
1. //astro.cpp  
2. oggetti.inserisci(Oggetto("etichetta", 70, -37));  
3. oggetti.inserisci(Oggetto("vetrina", 19, -37));  
4. oggetti.inserisci(Oggetto("astronave", 90, 38));  
5. oggetti.inserisci(Oggetto("equipaggio", 91, 38));
```

5.12.5 Dichiarazione azioni

Dichiarazione dei metodi nel file astro.h

```
1. //astro.h  
2. void azione_126(); // Lettura etichetta  
3. void azione_127(); // Apertura vetrina  
4. void azione_128(); // Chiusura vetrina  
5. void azione_129(); // Leggi libri  
6. void libro();
```

```
7. void scadenza();
```

5.12.5.1 leggi etichetta

```
1. //astro.cpp
2. void Astro::azione_126() {
3.     interfaccia.scrivi("\n    ---- REGOLAMENTO -----");
4.     interfaccia.scrivi("\n\n1) È possibile prendere un libro in "
5.             "\nprestito e restituirlo entro e non oltre 3 ore");
6.     interfaccia.scrivi("\n2) La restituzione è valida solo se i libri "
7.             "\nvengono lasciati al posto originario.");
8.     interfaccia.scrivi("\n3) Si consiglia di non lasciare i libri sparsi"
9.             "\n\nin biblioteca");
10.    interfaccia.scrivi("\n4) Si prega infine di fare silenzio!");
11.    interfaccia.scrivi("\n\nSaluti dal Bibliotecario... \n");
12. }
```

5.12.5.2 apri vetrina

```
1. //astro.cpp
2. void Astro::azione_127() {
3.     interfaccia.scrivi("la vetrina è aperta!\n");
4.     luogo_attuale = 38;
5. }
```

5.12.5.3 chiudi vetrina

```
1. //astro.cpp
2. void Astro::azione_128() {
3.     interfaccia.scrivi("la vetrina è chiusa!\n");
4.     luogo_attuale = 37;
5. }
```

5.12.5.4 leggi libro

```
1. //astro.cpp
2. void Astro::azione_129() {
3.     if (oggetti.get Oggetto(og).get_luogo() == 0) { //SE È TRASPORTATO
4.         cout << "leggo: " << og;
5.         libro();
6.     } else interfaccia.scrivi("- Non ce l'hai.\n"); //astro.cpp
```

5.12.5.4.1 libro()

L'azione 129 (leggi libro) richiama la funzione libro() definita come segue:

```
1. //astro.cpp
2. void Astro::libro() {
3.     switch (oggetti.get Oggetto(og).get codice()) {
```

```

4.         case 90:
5.             cout << "\n|-----";
6.             cout << "\n|          ASTRONAVE";
7.             cout << "\n| Un'astronave (detta anche nave | L'umanita' non ha mai costruit
8.             o          stellare) e' un veicolo spaziale | un'autentica astronave, molti
9.             |          progettato per il viaggio interstellare | scienziati (Freeman Dyson e il
10.            o          cioe' in grado di raggiungere sistemi | Progetto Orione) hanno discuss
11.            e          stellari diversi da quello di partenza. | diverse ipotesi ingegneristiche
12.            |          La fantascienza (in particolare il filone | riguardanti la possibilita' di
13.            aggi        della space opera) abbonda di storie che | intraprendere in futuro dei viaggi
14.            |          descrivono questo genere di veicoli | interstellari.
15.            |          degli anni molti autori hanno descritto |;
16.            |          e nel corso astronavi di varie forme: |;
17.            |          in alcuni anime l'astronave principale |;
18.            |          corazzata giapponese della seconda guerra |;
19.            |          aveva la forma di una mondiale, nella |;
20.            |          Guida galattica per gli autostoppisti ci |;
21.            |          sono astronavi a forma di mattoni |;
22.            |          giallim mentre le classiche astronavi |;
23.            |          alien degli anni cinquanta sono i celebri |;
24.            |          dischi volanti. |;
25.            |          cout << "\n|-----";
26.            system("pause");
27.            break;
28.        case 91:
29.            cout << "\n|-----";

```

Inserimento all'interno dello switch-case dei casi per eseguire le azioni precedentemente descritte:

```
1. //astro.cpp
2. case 126:
3.     azione_126();
4.     break;
5. case 127:
6.     azione_127(); //CHIAMA AZIONE DI APERTURA VETRINA
7.     break;
8. case 128:
9.     azione_128(); //CHIAMA AZIONE DI CHIUSURA VETRINA
10.    break;
```

```
11. case 129:  
12.     azione_129();  
13.     break;
```

5.12.6 Implementazione metodi per funzionamento biblioteca

5.12.6.1 *prendi_specifiche()*

Il metodo prendi_specifiche() permette di prendere un oggetto libro all'interno della biblioteca ed inserirlo in lista.

```
1. //astro.cpp  
2. bool Astro::prendi_specifiche() {  
3.     bool problemi = true;  
4.     if (og == 4 && oggetti.get_oggetto(22).get_luogo() == 0) {  
5.         interfaccia.scrivi("Togli prima il camice.");  
6.         stringa_risposta = "ti e' stato detto di togliere prima il camice."; //Modifica PMF(storia)  
7.         storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)  
8.     } else if (og == 22 && oggetti.get_oggetto(4).get_luogo() == 0) {  
9.         interfaccia.scrivi("Togli prima la tuta.");  
10.        stringa_risposta = "ti e' stato detto di togliere prima la tuta."; //Modifica PMF(storia)  
11.        storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)  
12.    } else if (luogo_attuale == 37 || luogo_attuale == 38) {  
13.        Lista < int > ::posizione p = libriInPrestito.primolista();  
14.        for ( int j = 0;  
15.            (!libriInPrestito.finelista(p)) && (j < MAX_LIBRI)); p = libriInPrestito.succ  
16.            clista(p)) {}  
17.        Oggetto oggettoPreso = oggetti.get_oggetto(og);  
18.        int codOggetto = oggettoPreso.get_codice();  
19.        if (codOggetto >= 90) {  
20.            libriInPrestito.inslista(codOggetto, p);  
21.        }  
22.        system("cls");  
23.        cout << "Hai preso in prestito " << oggettoPreso.get_nome() << ".\n";  
24.        oggetti.set_luogo(og, 0);  
25.    }  
26.    problemi = false;  
27.    return problemi;  
28. }
```

5.12.6.2 lascia_specifiche()

Il metodo lascia_specifiche() permette di lasciare un oggetto libro controllando che sia presente nella lista dei libri in prestito, e nel caso in cui fosse presente permette di cancellarlo dalla lista.

```
1. bool Astro::lascia_specifiche() {
2.     /*bool lasciato = true;                                if (luogo_attuale >= 9) {          o
3.     oggetti.set_luogo(og, -99);                      interfaccia.scrivi("Si e' perso nello spazio.");
4.     stringa_risposta = "si e' perso nello spazio!!"; //Modifica PMF(storia)
5.     storia_gioco.insStoria(stringa_comando , stringa_risposta); //Modifica PMF(storia)
6.     } else                                              lasciato = false;                  return lasciato;*/ //modifica Gallone
7.     bool lasciato = true; //Inizio Modifica Gallone - aggiorno il metodo per integrare le nuove funzionalità
8.     riferimento_tasca = tasca.primolista();
9.     riferimento_zaino = zaino_frigo.primolista();
10.    if (luogo_attuale == 9 || luogo_attuale == 10 || luogo_attuale == 11 || (luogo_attuale == 7 & parete_stagna_aperta == 1)) {
11.        if (og >= 35 && og <= 37) {
12.            while (!tasca.finelista(riferimento_tasca)) {
13.                if (oggetti.get Oggetto(tasca.leggilista(riferimento_tasca)).get Codice() == oggetti.get Oggetto(og).get Codice()) tasca.canclista(riferimento_tasca);
14.            else riferimento_tasca = tasca.succlista(riferimento_tasca);
15.        }
16.    }
17.    } //Fine modifica Gallone
18.    oggetti.set_luogo(og, -9999);
19.    interfaccia.scrivi("Si e' perso nello spazio.");
20.    } else if (luogo_attuale == 37 || luogo_attuale == 38) { //CONTROLLO BIBLIOTECA E VETRINA SUGLI OGGETTI CHE SI PERDONO
21.        if (luogo_attuale == 38) {
22.            int codOggetto = oggetti.get Oggetto(og).get Codice();
23.            if (codOggetto >= 90) // SE OGGETTO È UN LIBRO
24.            {
25.                Lista < int > :posizione p = libriInPrestito.primolista();
26.                for ( int j = 0;
27.                    (!libriInPrestito.finelista(p)) && (p < MAX_LIBRI)); p = libriInPrestito.succlista(p)) {
28.                    if (libriInPrestito.leggilista(p) == codOggetto) {
29.                        scadenze[j] = 30;
```

```

30.                     libriInPrestito.canclista(p);
31.                     oggetti.set_luogo(og, luogo_attuale);
32.                     cout << "Fatto.\nhai restituito " << oggetti.get_oggetto(og).get_nome() <
   < ".\n"; // scadenze[p]=30;
33.                     p = libriInPrestito.succlista(p);
34.                     j++;
35.                 }
36.             }
37.         }
38.     }
39. } else lasciato = false;
40. return lasciato;
41. }

```

5.12.6.3 scadenza()

Il metodo scadenza() permette di effettuare automaticamente la restituzione dei libri nel caso in cui l'utente si dimenticasse di restituire l'oggetto libro.

```

1. //astro.cpp
2. void Astro::scadenza() {
3.     Lista < int > ::posizione p = libriInPrestito.primolista();
4.     for ( int j = 0;
5.           (!libriInPrestito.finelista(p)) && (j < MAX_LIBRI)); p = libriInPrestito.succlista(p)) {
6.         int oggetto = libriInPrestito.legglista(p);
7.         int libro = oggetto - 90;
8.         int scadenza = --scadenze[libro];
9.         if (scadenza == 10) {
10.             for ( i = 1; i <= oggetti.get_n_oggetti(); i++) {
11.                 if (oggetti.get_oggetto(i).get_codice() == oggetto) {
12.                     cout << "AVVISO: Mancano 10min per la restituzione del libro " << "
   \n
   Corri in Biblioteca!\n";
13.                     break;
14.                 }
15.             }
16.         } else if (scadenza == 0) {
17.             scadenze[libro] = 100;
18.             for ( i = 1; i <= oggetti.get_n_oggetti(); i++) {
19.                 if (oggetti.get_oggetto(i).get_codice() == oggetto) {
20.                     oggetti.set_luogo(i, 38);
21.                     cout << "\nIl " << oggetti.get_oggetto(i).get_nome() << " e' tornato
   in Biblioteca. \n\n";

```

```

22.           cout << "\nA causa della mancata restituzione hai perso 10 minuti d
   el tuo tempo.";
23.           break;
24.       }
25.   }
26.   libriInPrestito.canclista(p);
27.   tempo = tempo - 10; //PENALITA'
28. }
29. j++;
30. }
31. }
```

5.12.6.4 aggiorna_specifiche()

Il metodo aggiorna_specifiche() permette l'aggiornamento ovvero il decremento della scadenza ad ogni azione.

```

1. //gioco.h // VIRTUAL
2. virtual void setCabina(int); /*
3. virtual void setParete(int); /*
4. virtual int getCabina(); /*
5. virtual int getParete(); /*
6. virtual void aggiorna_specifiche(); //biblioteca
7. virtual bool esegui_specifiche(int, Mappa & );
```

```

1. //gioco.cpp
2. void Gioco::aggiorna_specifiche() { }
```

```

1. //astro.h
2. #define MAX_LIBRI 100
3. #define TEMPO 50
4. #include "Portafoglio.h" //MODIFICA D-R(Pisani):Portafoglio + Carta di Credito
5. #include "CartadiCredito.h" //MODIFICA D-
   R(Pisani):Portafoglio + Carta di Credito
6.
7. class Astro: public Gioco {
1.
2. public: Astro();
3. virtual~Astro();
4. int getParete();
5. int getCabina();
```

```

6. void setParete(int);
7. void setCabina(int);
8. void aggiorna_specifiche(); //biblioteca
9.
10. private:
11. //inizio modifiche biblioteca scatigna
12. int scadenze[10] = {TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO};
13. int i;
14. Lista <int>libriInPrestito;

```

*FIX: il vettore scadenze[] non era inizializzato compromettendo il funzionamento della funzione scadenze() dunque è stato settato con la costante TEMPO a 50 minuti.

```

1. //astro.cpp
2. void Astro::aggiorna_specifiche() {
3.     scadenza();
4. }

```

5.12.6.5 ciak(Mappa & M, ifstream &)

Aggiunta del metodo aggiorna_specifiche() in ciak(Mappa & M, ifstream &) nel file gioco.cpp

```

1. void Gioco::ciak(Mappa &M, ifstream&)
2. {
3.
4.     string comando;
5.     Luogo l;
6.     Stato s; /*
7.     mappa = M;
8.     bool si;
9.
10.
11.    //Pila <string>* p = new Pila <string>();
12.    Pila <stato_comando>* p = new Pila <stato_comando>();
13.
14.
15.    do
16.    {
17.        svuotaPila(*p);
18.        init();
19.        introduzione();
20.
21.        initPers(&insPersonaggi);
22.
23.        interfaccia.pausa(); // <SP>
24.        si = attiva_enigmi(); // MODIFICA D-R(D-R)
25.
26.        do // ciclo di gioco
27.        {
28.            //INIZIO MODIFICA MARCO DAMONE
29.            //RIMOZIONE DELLE SOTTOSTANTI LINEE DI CODICE
30.            //if(luogo_attuale==6 && mappaaperta())

```

```

31.          //luogo_attuale++;
32.          //FINE MODIFICA MARCO DAMONE
33.          if(mappa.get_nome_luogo(luogo_attuale) == "Luogotrasporto")
34.          {
35.              trasporto(luogo_attuale);
36.              stringa_risposta = "sei andato " + mappa.get_nome_luogo(luogo_attuale)
37.              + "."; //Modifica PMF(storia)
38.              storia_gioco.insStoria(stringa_comando, stringa_risposta);
39.          }
40.          l=mappa.get_luogo(luogo_attuale);
41.          if((!Luogo_p.pilavuota()) && (Codice_luogo.leggipila()!=luogo_attuale)) //MODI
42.          {
43.              cout<<"Provieni dal luogo: "<< Codice_luogo.leggipila() << ":" << Luogo
44.              _p.leggipila(); // MODIFICA POLIMENA
45.              Luogo_p.inpila(mappa.get_nome_luogo(luogo_attuale)); //MODIFICA POLIMENA
46.              Codice_luogo.inpila(luogo_attuale); //MODIFICA POLIMENA
47.              if(luogo_attuale == 7)
48.              {
49.                  if(oggetti.get_oggetto(4).get_luogo() != 0 || oggetti.get_oggetto(11).g
50.                      et_luogo()!= 0)
51.                      bacheca.NuovoMessaggioGioco("*non uscire fuori dall'astronave se no
52.                      n hai l'equipaggiamento da astronauta");
53.                  //else
54.                  //bacheca.CancellaMessaggioGioco("*non uscire fuori dall'astronave se n
55.                      on hai l'equipaggiamento da astronauta");
56.              }
57.              // MODIFICA D-R(Colturi):Luoghi a Pagamento-----
58.              if(luogo_attuale > 15) //|
59.              {
60.                  //|
61.                  //Luogo 1 = M.get_luogo(luogo_attuale); //|
62.                  //|
63.                  interfaccia.disegna_scena();
64.                  personaggio_entra(); // modifica zagaria
65.
66.                  //if ()
67.                  interfaccia.elenca_oggetti(oggetti.posizionati_in(luogo_attuale), "Vedo:");
68.
69.                  //else
70.                  aggiorna_premi(); //modifiche Francesco Cosma - Premi
71.                  aggiorna_specifiche(); //biblioteca
72.                  aggiorna_tempo();
73.                  comando = interfaccia.leggi_comando();
74.                  interpreta(comando,M,p,s,si);
75.                  aggiorna_progresso(); //modifiche Francesco Cosma - Premi
76.              }
77.              while(!fine_partita);
78.          }
79.      while (riparti);

```

5.12.7 ChangeLog

Problema: la lista dei libri in prestito aumenta di 3 elementi per ogni libro preso

Causa: il metodo Gioco.prendi_specifiche() in gioco.cpp per overloading punta al metodo Astro.prendi_specifiche() in Astro.cpp che incrementa ogni volta la lista dei libri in prestito.

Il metodo prendi_specifiche() di Gioco.cpp è il seguente:

```
1. bool Gioco::prendi_specifiche() {  
2.     return false;  
3. }
```

Soluzione: il metodo risulta essere utilizzato solo come !TRUE nelle condizioni del metodo Gioco.prendi() (riga 17 e riga 30) dunque ho ritenuto utile senza stravolgere il codice di definirlo senza corpo solo per farlo puntare al metodo omonimo in Astro.cpp.

```
4. bool Gioco::prendi_specifiche() { }
```

Il metodo è stato utilizzato come condizione soltanto una volta sola (riga 30), garantendo così il corretto funzionamento del metodo in Astro.

```
1. //gioco.cpp  
2. void Gioco::prendi() { //modifica Gallone Gianmarco - Riadattamento metodo prendi per tasca, zaino e portafoglio.  
3.     bool fatto = false;  
4.     bool cpz = true;  
5.     riferimento_tasca = tasca.primolista();  
6.     riferimento_zaino = zaino_frigo.primolista();  
7.     bool trovato = false;  
8.     while (!tasca.finelista(riferimento_tasca)) {  
9.         if (oggetti.get_oggetto(og).get_codice() == oggetti.get_oggetto(tasca.leggilista(riferimento_tasca)).get_codice()) trovato = true;  
10.        riferimento_tasca = tasca.succlista(riferimento_tasca);  
11.    }  
12.    while (!zaino_frigo.finelista(riferimento_zaino)) {  
13.        riferimento_zaino = zaino_frigo.succlista(riferimento_zaino);  
14.    }  
15.    if (oggetti.get_oggetto(og).get_luogo() == 0) interfaccia.scrivi("- Gia' fatto.");  
16.    else if (oggetti.get_oggetto(og).get_luogo() < 0) interfaccia.scrivi("- Non e' possibile.");  
17.    //else if ((!prendi_specifiche() && (og == 27) || !prendi_specifiche() && (og >= 36 && og <= 44)))  
18.    else if ((og == 27) || (og >= 36 && og <= 44)) {  
19.        bool port = false;  
20.        port = portafoglio.hai_Portafoglio(oggetti);  
21.        if (port && (og >= 36 && og <= 44)) {
```

```

22.         if (!fatto) portafoglio.Prendi_Banconota(oggetti, og, interfaccia);
23.         oggetti.set_luogo(og, 0);
24.     } else if (!port && (og >= 36 && og <= 44)) interfaccia.scrivi(" Non e' Possibile prendere denaro se non hai gia' preso il portafoglio !");
25.     else if (!port && (og == 27)) interfaccia.scrivi("Non e' Possibile prendere la carta di credito se non hai gia' preso il portafoglio !");
26.     else if (port && (og == 27)) {
27.         oggetti.set_luogo(og, 0);
28.         interfaccia.scrivi("Presal! L'hai nel portafoglio");
29.     }
30. } else if (!prendi_specifiche()) {
31.     if (og >= 77 && og <= 79) {
32.         oggetti.set_luogo(og, 0);
33.         tasca.inslista(og, riferimento_tasca);
34.         interfaccia.scrivi("Inserito nella tasca!");
35.     } else if (og >= 67 && og <= 76) {
36.         if (tasca.listavuota()) {
37.             interfaccia.scrivi("Non hai buoni pasto per acquistarlo!");
38.             cpz = false;
39.         } else {
40.             interfaccia.scrivi("Dovresti comprare le pietanze, non prenderle...");
41.             cpz = false;
42.         } // else{ oggetti.set_luogo(og,0); // zaino_frigo.inslista(riferimento_zaino,og);
// interfaccia.scrivi("Inserito nello zaino!");}
43.     } else oggetti.set_luogo(og, 0); //Fine Modifica Gallone
44.     if (!preso_specifiche() && (cpz)) // MANCA NELLE VERSIONI INTERMEDI
45.         interfaccia.scrivi("Fatto.");
46.     } // modifica Gallone - spostamento parentesi per correzione di un errore che si generava nel momento in cui // si prendeva un oggetto che non poteva esser preso
47.     bacheca.CancellaMessaggioGioco("*non uscire fuori dall'astronave se non hai l'equipaggiamento o da astronauta");
48. }

```

Problema: errore di compilazione in caricastatodialogo(string nome)

Causa: nel metodo **caricastatodialogo(string nome)** il controllo di esistenza del file Statidialoghi.txt in gioco.cpp causa un errore di compilazione (riga 5).

Soluzione: il controllo è stato sostituito con un'altra condizione (riga 6) che ha lo stesso effetto e non causa errori.

```

1. //gioco.cpp
2. int Gioco::caricastatodialogo(string nome) {
3.     int stato = 0;

```

```
4.     string rigafile;
5.     ifstream filestati("Statidialoghi.txt", ios:: in );
6.     //if (!filestati == NULL) {
7.     if (!filestati.is_open()) {
8.         cout << "File Statidialoghi.txt non trovato";
9.     } else {
10.         while (filestati.get() != '?') {
11.             getline(filestati, rigafile);
12.             leggistato( & stato, nome, rigafile); //legge lo stato all'interno
13. della riga del file
14.     }
15. }
16. filestati.close();
17. return stato;
18. }
```

Correzione Funzionalità lascia libro (Andrea Tursi)

È stato riscontrato un errore nel lasciare i due libri presi in prestito dalla vetrina della biblioteca. L'errore causava l'interruzione improvvisa del gioco.

```
Hai preso in prestito astronave.  
Fatto.  
Sei vicino alla vetrina.  
Vedo:  
- equipaggio.  
Tempo residuo: 292  
Tempo aggiuntivo dovuto allo Stato di Salute: 2  
Tempo residuo: 286  
Cosa devo fare?  
Lascia astronave  
  
Process returned -1073741819 (0xC0000005) execution time : 74.662 s  
Press any key to continue.
```

Il problema è stato risolto andando a modificare il file Astro.cpp, dove nel metodo **lascia_specifiche()**, alla riga 1242, veniva assegnato il valore intero 30 alla posizione p-esima (con p di tipo posizione lista) del vettore scadenze.

Assegnazione che veniva ripetuta correttamente alla riga 1247, utilizzando l'indice j creato appositamente.

```
1240         if (libriInPrestito.legglista(p) == codOggetto)  
1241         {  
1242             scadenze[libriInPrestito.legglista(p)] = 30;  
1243             libriInPrestito.canclista(p);  
1244             oggetti.set_luogo(og, luogo_attuale);  
1245             cout << "Fatto.\nhai restituito "  
1246                 << oggetti.get_oggetto(og).get_nome() << ".\n";  
1247             scadenze[j]=30;  
1248             p = libriInPrestito.succlista(p);  
1249             j++;  
1250         }
```

```
1240         if (libriInPrestito.legglista(p) == codOggetto)  
1241         {  
1242             scadenze[j] = 30;  
1243             libriInPrestito.canclista(p);  
1244             oggetti.set_luogo(og, luogo_attuale);  
1245             cout << "Fatto.\nhai restituito "  
1246                 << oggetti.get_oggetto(og).get_nome() << ".\n";  
1247             p = libriInPrestito.succlista(p);  
1248         }
```

```
Hai preso in prestito astronave.  
Fatto.  
  
Sei vicino alla vetrina.  
Vedo:  
- equipaggio.  
Tempo residuo: 316  
Tempo aggiuntivo dovuto allo Stato di Salute: 1  
Tempo residuo: 312  
Cosa devo fare?  
lascia astronave  
  
Fatto.  
hai restituito astronave.  
  
Sei vicino alla vetrina.  
Vedo:  
- astronave;  
- equipaggio.  
Tempo residuo: 311  
Tempo aggiuntivo dovuto allo Stato di Salute: 1  
Tempo residuo: 307  
Cosa devo fare?
```

5.13 IMPLEMENTAZIONE BARI – ROMA – PISA

L'integrazione del progetto di Narracci in quello di Tursi (progetto base) ha portato a delle modifiche necessarie e che saranno spiegate in questo capitolo.

Per l'inserimento dei nuovi luoghi e dei nuovi oggetti, sono stati modificati i seguenti file:

- Mappa.nav
- Descrizioni (cartella)
- Astro.cpp
- Asto.h
- Gioco.cpp
- Gioco.h
- Luogo.cpp
- Luogo.h
- Oggetti.cpp
- Oggetti.h
- Oggetto.cpp
- Oggetto.h
- Veicolo.cpp
- Veicolo.h

Per la corretta importazione del progetto di Narracci, sono stati trasportati nel progetto base i seguenti file:

- Bagagliaio.cpp
- Bagagliaio.h

- Autobus.cpp
- Autobus.h
- Automobile.cpp
- Automobile.h

5.13.1 Mappa.nav

In questo file sono state effettuate le seguenti modifiche:

- 45,Roma,474600000000,via 6,8,8
- 46,Bari,450000130000,via 6,8,8
- 47,Pisa,004500000000,via 6,8,8

Identificati da:

- **Codice** (45, 46, 47)
- **Nome del luogo**
- **Codice direzioni** (NNSEEEUUDD)
- **Tipo di strada** (via 6,8,8)

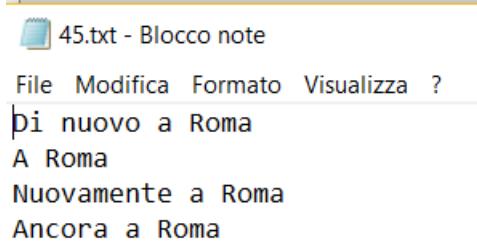
È stato modificato il codice direzioni al luogo 13 “**Stazione di servizio**” per arrivare a **Bari** digitando *Est*:

- Narracci: 000046000004

5.13.2 Descrizioni(cartella)

Nella cartella Descrizioni sono stati importati tre file .txt dal progetto di Narracci.

Ne riporto solo un esempio.



5.13.3 Astro.cpp

In questo file sono state apportate le seguenti modifiche:

- Inserimento vocaboli

```
//INIZIO modifiche DELLA FOLGORE GRAZIA
    vocabolario.inserisci("ticket_bus", 114);
    vocabolario.inserisci("autobus", 115);
    vocabolario.inserisci("chiave_auto", 116);
```

```
    vocabolario.inserisci("automobile", 117);
//FINE modifiche DELLA FOLGORE GRAZIA
```

- Inserimento azioni

```
azioni.inserisci(1117, 80); //Modifica DELLA FOLGORE GRAZIA
```

- Inserimento oggetti

```
//INIZIO modifiche DELLA FOLGORE GRAZIA

oggetti.inserisci(Oggetto("un ticket_bus", 114, 5));
oggetti.inserisci(Autobus("un autobus", 115, 13, 20, false));
oggetti.inserisci(Oggetto("una chiave_auto", 116, 5));
oggetti.inserisci(Oggetto("un'automobile", 117, 13));

//FINE modifiche DELLA FOLGORE GRAZIA
```

- Controllo ticket prima di prendere l'autobus Astro::bool prendi_specifiche()

```
//INIZIO modifiche DELLA FOLGORE GRAZIA

else if (og == 117 && oggetti.get_oggetto(116).get_luogo() != 0){ //se non possiedi il ticket_bus
(autobus->117)

    interfaccia.scrivi("Devi prendere un ticket_bus.");
    stringa_risposta = "ti e' stato detto di prendere un ticket_bus.";
    storia_gioco.insStoria(stringa_comando , stringa_risposta);

}

else if (og == 117 && oggetti.get_oggetto(116).get_luogo() == 0){ //se possiedi il ticket_bus

    if (automobile.get_inUso()==false){

        interfaccia.scrivi("Sei salito sull'autobus.");
        oggetti.set_luogo(117, 0);      //sei sull'autobus
        autobus.set_inUso(true);
        problemi = false;
        stringa_risposta = "sei salito sull'autobus.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);

    }else{
```

```

        interfaccia.scrivi("Devi prima lasciare l'automobile");
        stringa_risposta = "Ti e' stato detto di lasciare prima l'automobile.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }
}

```

- Controllo chiavi prima di prendere l'automobile Astro::bool prendi_specifiche()

```

else if (og == 119 && oggetti.get_oggetto(118).get_luogo() != 0){// se non possiedi le chiavi dell' auto (auto> 119)

    interfaccia.scrivi("Devi prendere la chiave.");
    stringa_risposta = "ti e' stato detto di prendere la chiave.";
    storia_gioco.insStoria(stringa_comando , stringa_risposta);
}

else if (og == 119 && oggetti.get_oggetto(118).get_luogo() == 0){// se possiedi le chiavi dell' auto

    if (autobus.get_inUso()==false){

        interfaccia.scrivi("Sei salito sull'automobile.");
        oggetti.set_luogo(119, 0);
        automobile.set_inUso(true);
        problemi = false;
        stringa_risposta = "Sei salito sull'automobile.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);

    }else{

        interfaccia.scrivi("Devi prima lasciare l'autobus");
        stringa_risposta = "Ti e' stato detto di lasciare prima l'autobus.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }
}
//FINE modifiche DELLA FOLGORE GRAZIA

```

- Controllo se il veicolo può essere lasciato nel luogo corrente Astro::bool lascia_specifiche()

```

bool lascia_specifiche():

//INIZIO modifiche DELLA FOLGORE GRAZIA

```

```

if (luogo_attuale == 13 || (mappa.get_nome_luogo(luogo_attuale) == "Bari") ||
(mappa.get_nome_luogo(luogo_attuale) == "Roma") || (mappa.get_nome_luogo(luogo_attuale) ==
"Pisa")){
    if (og == 117){//autobus

        oggetti.set_luogo(og, luogo_attuale);

        autobus.set_inUso(false);

        interfaccia.scrivi("Hai lasciato l'autobus");

        stringa_risposta = "hai lasciato l'autobus";

        storia_gioco.insStoria(stringa_comando , stringa_risposta);

    }else if(og == 119){//automobile

        oggetti.set_luogo(og, luogo_attuale);

        automobile.set_inUso(false);

        interfaccia.scrivi("Hai lasciato l'automobile");

        stringa_risposta = "hai lasciato l'automobile";

        storia_gioco.insStoria(stringa_comando , stringa_risposta);

    }

}

else{
    lasciato = false;

    if(og == 117){

        interfaccia.scrivi("Non puoi scendere dall'autobus qui");

        stringa_risposta = "hai tentato di scendere dall'autobus dove non potevi";

        storia_gioco.insStoria(stringa_comando , stringa_risposta);

    }

    if(og == 119){

        interfaccia.scrivi("Non puoi scendere dall'auto qui");

        stringa_risposta = "hai tentato di scendere dall'auto dove non potevi";

        storia_gioco.insStoria(stringa_comando , stringa_risposta);

    }

}
}
} //FINE modifiche DELLA FOLGORE GRAZIA

```

- Inserimento azione “80” (*guarda bagagliaio*):

```
//INIZIO modifiche DELLA FOLGORE GRAZIA
```

```

void Astro ::azione_80(){

if((automobile.get_inUso()==false)&&(autobus.get_inUso()==false)){

    bool controllo = false;

    bool conferma = false;

    bool uscita = false;

    string input_utente;

    int selezione;

    interfaccia.scrivi("L'automobile dispone di un bagagliaio");

    stringa_risposta = "hai notato che l'auto dispone di un bagagliaio";

    storia_gioco.insStoria(stringa_comando , stringa_risposta);

    interfaccia.scrivi_parziale("Vuoi utilizzarlo per riporvi/prelevare oggetti? [si/no]");

    while(controllo!=true){

        interfaccia.a_capo();

        cin >> input_utente;

        if(input_utente=="S" || input_utente=="s" || input_utente=="Si" || input_utente== "SI" || input_utente=="si"){

            conferma=true;           //Il giocatore conferma di voler usare il bagagliaio

            controllo=true;

        }

        else if(input_utente=="N" || input_utente=="n" || input_utente=="No" || input_utente== "NO" || input_utente=="no"){

            conferma=false;          //Il giocatore conferma di non voler usare il bagagliaio

            controllo=true;

        }

        else{   interfaccia.scrivi("Non capisco.");

                aggiorna_tempo();

                interfaccia.a_capo();

                interfaccia.scrivi("L'automobile dispone di un bagagliaio");

                interfaccia.scrivi_parziale("Vuoi utilizzarlo per riporvi/prelevare oggetti? [si/no]");

                interfaccia.a_capo();

            }

    }

    if(conferma){


```

```

storia_gioco.insStoria(stringa_comando , "hai deciso di usare il bagagliaio dell'auto");
system("cls");

if(oggetti.get Oggetto(118).get_luogo() == 0){

    interfaccia.scrivi("Puoi utilizzare il bagagliaio perche' possiedi la chiave");

    interfaccia.scrivi("Seleziona:");

    interfaccia.scrivi(" 0: per depositare gli oggetti");
    interfaccia.scrivi(" 1: per prelevare gli oggetti");
    interfaccia.scrivi(" 2: per vedere il contenuto del bagagliaio");
    interfaccia.scrivi(" 3: per vedere gli oggetti che possiedi");
    interfaccia.scrivi(" 4: chiudere il bagagliaio");

    interfaccia.a_capo();

}else{

    interfaccia.scrivi("Devi prendere la chiave dell'auto");
    uscita=true;
}

while(uscita!=true){

    int num_ogg_inventario = oggetti.posizionati_in(0).get_n_oggetti();
    interfaccia.scrivi_parziale("Cosa vuoi fare:");
    cin >> selezione;

    switch(selezione){

        case 0:

            storia_gioco.insStoria("0" , "hai deciso di depositare i tuoi oggetti nel
bagagliaio");

            if(num_ogg_inventario>0){

                for(int i=1; i<=num_ogg_inventario; i++){

                    //L'oggetto viene depositato se non è una chiave

                    if((oggetti.posizionati_in(0).get Oggetto(i).get codice())!=116){           //116 -> Chiave Automobile
                        automobile.bagagliaio.depositaBagaglio(oggetti.posizionati_in(0).get Oggetto(i));      //L'oggetto
                    }
                }
            }
        }
    }
}

```

```

        }

    }

    oggetti.set_luogo_oggetti_pos_in(0, -99);

}

interfaccia.a_capo();

interfaccia.scrivi("Oggetti depositati nel bagagliaio");

interfaccia.a_capo();

uscita = true;

break;

case 1:

    storia_gioco.insStoria("1", "hai deciso di prelevare i tuoi oggetti dal
bagagliaio");

    if(!automobile.bagagliaio.bagagliaioVuoto()){

        interfaccia.a_capo();

        interfaccia.scrivi("Oggetti prelevati dal bagagliaio");

    }else{

        interfaccia.a_capo();

        interfaccia.scrivi("Non hai prelevato nulla: il bagagliaio e' vuoto");

    }

    while(!automobile.bagagliaio.bagagliaioVuoto()){

        Oggetto oggetto = automobile.bagagliaio.estraiBagaglio();

        int indice =
oggetti.get_oggetto_indice_by_codice(oggetto.get_codice());

        oggetti.set_luogo(indice, 0);

    }

    uscita = true;

    break;

case 2:

    storia_gioco.insStoria("2", "hai deciso di vedere il contenuto del
bagagliaio");

    interfaccia.a_capo();

    automobile.bagagliaio.vediBagagli();

    interfaccia.a_capo();

```

```

        uscita = true;
        break;

    case 3:
        storia_gioco.insStoria("3" , "hai deciso di vedere gli oggetti che possiedi");
        interfaccia.a_capo();
        interfaccia.elenca_oggetti(oggetti.posizionati_in(0),"Possiedi:");
        if (ritirato == true){
            interfaccia.scrivi("Possiedi come documento: ");
            primo_doc.elenca_documenti(0);
        }
        uscita = true;
        break;

    case 4:
        storia_gioco.insStoria("4" , "hai deciso di chiudere il bagagliaio dell'auto");
        uscita = true;
        break;

    default:
        system("cls");
        interfaccia.scrivi("Non capisco.");
        aggiorna_tempo();
        interfaccia.a_capo();
        interfaccia.scrivi("Seleziona:");
        interfaccia.scrivi(" 0: per depositare gli oggetti");
        interfaccia.scrivi(" 1: per prelevare gli oggetti");
        interfaccia.scrivi(" 2: per vedere il contenuto del bagagliaio");
        interfaccia.scrivi(" 3: per vedere gli oggetti che possiedi");
        interfaccia.scrivi(" 4: chiudere il bagagliaio");
        interfaccia.a_capo();
        break;
    }

}
}

```

```

}else{
    if(automobile.get_inUso()){
        interfaccia.scrivi("Non noti nulla dall'interno dell'auto");
        storia_gioco.insStoria(stringa_comando , "non hai notato nulla dall'interno");
    }
    if(autobus.get_inUso()){
        interfaccia.scrivi("Non noti nulla dall'autobus");
        storia_gioco.insStoria(stringa_comando , " non hai notato nulla dall'autobus");
    }
}
//FINE modifiche DELLA FOLGORE GRAZIA

```

- Inserimento azione 80 in Astro::bool esegui_specifiche()

```

case 80:
    azione_80(); //Modifica DELLA FOLGORE GRAZIA - Guarda bagagliaio
    break;

```

5.13.4 Astro.h

- Prototipo azione_80 “guarda automobile”

```

void azione_80(); //Modifica DELLA FOLGORE GRAZIA - Guarda automobile

```

5.13.5 Gioco.cpp

- Inserimento di un controllo che verifica se le direzioni digitare dal giocatore sono ammesse

```

bool Gioco::direzioni(Mappa &M) {
    //INIZIO modifiche DELLA FOLGORE GRAZIA
    bool cambia_dir = true;
    int a = mappa.luogo_adiacente(luogo_attuale,cod_parola1);
    if (a == 0){
        interfaccia.scrivi("- Di li' non puoi andare");
        stringa_risposta = "di li' non sei potuto andare."; //Modifica PMF(storia)
    }
}

```

```

    storia_gioco.insStoria(stringa_comando , stringa_risposta); //Modifica PMF(storia)
    cambia_dir = false; //PANNO

}else{

    if (((autobus.get_inUso()==true)|| (automobile.get_inUso()==true)) && (a<45 && a!=13)) {

        if(autobus.get_inUso()==true){

            interfaccia.scrivi("- Non puoi andare li' con l'autobus");

            stringa_risposta = "hai provato ad andare con l'autobus in posti non
autorizzati";

            storia_gioco.insStoria(stringa_comando , stringa_risposta);

        }

        if(automobile.get_inUso()==true){

            interfaccia.scrivi("- Non puoi andare li' con l'auto");

            stringa_risposta = "hai provato ad andare con l'auto in posti non
autorizzati";

            storia_gioco.insStoria(stringa_comando , stringa_risposta);

        }

        cambia_dir = false; //Modifica Rosita Galiandro

    }else{

        int b = a - 9; //Aggiustamento per ricerca luogo da file DELLA FOLGORI GRAZIA

        if((a>=45) && (a<=47) && ((mappa.get_nome_luogo(b) == "Bari") || (mappa.get_nome_luogo(b) ==
"Roma") || (mappa.get_nome_luogo(b) == "Pisa"))){

            a = b;

            if((autobus.get_inUso()==false)&&(automobile.get_inUso()==false)){

                interfaccia.scrivi("- Impieghi piu' tempo muovendoti a piedi");

                tempo = tempo - 3;

            }

        }

        luogo_attuale = a;

        if(mappa.get_nome_luogo(a) != "Luogotrasporto"){

            stringa_risposta = "sei andato " + mappa.get_nome_luogo(luogo_attuale) + ". "; //Modifica
PMF(storia)

            storia_gioco.insStoria(stringa_comando , stringa_risposta); //Modifica PMF(storia)

        }

    }

}

```

```

        }
    }

    return cambia_dir;
    //FINE modifiche DELLA FOLGORE GRAZIA
}

```

5.13.6 Gioco.h

- Inclusione dei nuovi file

```
#include "Automobile.h" //Modifica DELLA FOLGORE GRAZIA
#include "Autobus.h" //Modifica DELLA FOLGORE GRAZIA
```

- Creazione dei veicoli

```
//INIZIO modifiche DELLA FOLGORE GRAZIA
Autobus autobus;
Automobile automobile;
//FINE modifiche DELLA FOLGORE GRAZIA
```

5.13.7 Luogo.cpp

- Funzione leggi_predefinito che controlla i luoghi accessibili con veicoli

```
//INIZIO modifiche DELLA FOLGORE GRAZIA
bool Luogo::leggi_predefinito() {
    if (id>=45)
        return true;
    else
        return false;
}

//FINE modifiche DELLA FOLGORE GRAZIA
```

5.13.8 Luogo.h

- Prototipo funzione di controllo

```
bool leggi_predefinito(); //Modifica DELLA FOLGORE GRAZIA
```

5.13.9 Oggetti.cpp

- Aggiunta metodo che permette di cambiare il posizionamento di un sottoinsieme di oggetti

```
//INIZIO modifiche DELLA FOLGORE GRAZIA

void Oggetti::set_luogo_oggetti_pos_in(int l_or, int l_dest){

    for (int i = 1; i <= fo; i++) {
        if (abs(oggetti[i].get_luogo()) == l_or){
            if(oggetti[i].get_codice()!=116 ){           //se l'oggetto è diverso dalla chiave
dell'automobile
                oggetti[i].set_luogo_n(l_dest);
            }
        }
    }

//FINE modifiche DELLA FOLGORE GRAZIA
```

5.13.10 Oggetti.h

- Prototipo metodo per cambio posizione oggetti

```
void set_luogo_oggetti_pos_in(int, int); //Modifica DELLA FOLGORE GRAZIA
```

5.13.11 Oggetto.h

```
void set_luogo_n(int);      //Modifica DELLA FOLGORE GRAZIA
void set_isVeicolo(bool); //Modifica DELLA FOLGORE GRAZIA
bool is_Veicolo();   //Modifica DELLA FOLGORE GRAZIA
bool veicolo; //Modifica DELLA FOLGORE GRAZIA
```

5.13.12 Oggetto.cpp

```
void Oggetto::set_luogo_n(int lg){ //Modifica DELLA FOLGORE GRAZIA
    luogo = lg;
}

void Oggetto::set_isVeicolo(bool b){      //Modifica DELLA FOLGORE GRAZIA
    veicolo = b;
}
```

```

bool Oggetto::is_Veicolo(){           //Modifica DELLA FOLGORE GRAZIA
    return veicolo;}
```

5.13.13 Veicolo.h

```

#ifndef H_VEICOLO
#define H_VEICOLO

#include "Oggetto.h"
/* DEFINIZIONE DELLA CLASSE VEICOLO*/

using namespace std;

class Veicolo: public Oggetto {

public:

    //INIZIO MODIFICHE PANNO
    Veicolo();
    Veicolo(string, int, int, unsigned short, bool);
    ~Veicolo();
    void set_numPosti(unsigned short);
    void set_inUso(bool);
    unsigned short get_numPosti();
    bool get_inUso();
    //FINE MODIFICHE PANNO

    void pieno_benzina();
    void aggiorna_carburante();
    void set_indice_di_consumo(int);
    int get_livello_benzina();
    void set_livello_benzina(int);

private:

    int indice_di_consumo;
    unsigned short livello_benzina;
    unsigned short numPosti;      //Numero posti - MODIFICA PANNO
    bool inUso;                  //Indica se il veicolo è in uso - MODIFICA PANNO

};
```

#endif

5.13.14 Veicolo.cpp

```
/* METODI DELLA CLASSE VEICOLO */
#include "Veicolo.h"
#include "Oggetto.h"

void Veicolo::pieno_benzina() {
    set_livello_benzina(20);
}

int Veicolo::get_livello_benzina()
{   return livello_benzina;}

void Veicolo::set_indice_di_consumo(int i)
{
    indice_di_consumo = i;
}

void Veicolo::aggiorna_carburante()
{
    // m*indice di consumo Ã¨ il consumo per ogni metro percorso
    // la distanza complessima percorsa Ã¨ data dal navigatore

    livello_benzina=livello_benzina-indice_di_consumo;
}

void Veicolo::set_livello_benzina(int l)
{
    livello_benzina=l;
}

//INIZIO MODIFICHE PANNO
Veicolo::Veicolo():Oggetto(){
    numPosti = 0;
    inUso = false;
    indice_di_consumo = 0;
    livello_benzina = 0;
    set_isVeicolo(true);
}

Veicolo::Veicolo(string n, int c, int l, unsigned short numP, bool inU):Oggetto(n, c, l){
    numPosti = numP;
    inUso = inU;
    indice_di_consumo = 0;
    livello_benzina = 0;
    set_isVeicolo(true);
}

Veicolo::~Veicolo(){

}

void Veicolo::set_numPosti(unsigned short numP){
    numPosti = numP;
```

```

}

void Veicolo::set_inUso(bool inU){
    inUso = inU;
}

unsigned short Veicolo::get_numPosti(){
    return numPosti;
}

bool Veicolo::get_inUso(){
    return inUso;
}

//FINE MODIFICHE PANNO

```

5.13.15 Bagagliaio.h

```

/* HEADER FILE CONTENENTE LA DEFINIZIONE DELLA
 * CLASSE BAGAGLIAIO IN USO ALL'AUTOMOBILE
 * I dettagli relativi alla specifica del tipo BAGAGLIAIO
 * sono riportati nella documentazione corrispondente.
 */

```

```

#ifndef BAGAGLIAIO_H_
#define BAGAGLIAIO_H_

#include "Oggetto.h"

//Dimensione massima
const unsigned short DIMMAX = 10;

//Definizione del tipo "bagaglio"
struct bagaglio{
    Oggetto oggetto;
    bool presente;
};

class Bagagliaio{
public:
    Bagagliaio();
    ~Bagagliaio();
    bool bagagliaioVuoto();
    bool bagagliaioPieno();
    void depositaBagaglio(Oggetto);
    void rimuoviBagaglio(Oggetto);
    Oggetto vediBagaglio(Oggetto);
    Oggetto estraiBagaglio();
    int cercaBagaglio(Oggetto);
    void vediBagagli();
private:

```

```

        int cercaScompartoLibero();
        bagaglio bagagli[DIMMAX];
        unsigned short num_bagagli;
    };

#endif /* BAGAGLIAIO_H_ */

```

5.13.16 Bagagliaio.cpp

```

#include "Bagagliaio.h"
#include <cstdlib>
#include <iostream>

using namespace std;

Bagagliaio::Bagagliaio(){
    for(int indice =0; indice<DIMMAX; indice=indice+1)
        bagagli[indice].presente = false;
    num_bagagli = 0;
}

Bagagliaio::~Bagagliaio(){

}

bool Bagagliaio::bagagliaioVuoto(){
    return(num_bagagli==0);
}

bool Bagagliaio::bagagliaioPieno(){
    return(num_bagagli==DIMMAX);
}

void Bagagliaio::depositaBagaglio(Oggetto ogg){
    int posizione = cercaScompartoLibero();
    if(posizione != -DIMMAX){
        bagagli[posizione].oggetto = ogg;
        bagagli[posizione].presente = true;
        num_bagagli++;
    }
}

void Bagagliaio::rimuoviBagaglio(Oggetto ogg){
    if(!bagagliaioVuoto()){
        int posizione = cercaBagaglio(ogg);
        if(posizione != -DIMMAX){
            bagagli[posizione].presente = false;
            num_bagagli--;
        }
    }
}

Oggetto Bagagliaio::vediBagaglio(Oggetto ogg){

```

```

        if(!bagagliaioVuoto()){
            int posizione = cercaBagaglio(ogg);
            if(posizione != -DIMMAX)
                return bagagli[posizione].oggetto;
        }
    }

Oggetto Bagagliaio::estraiBagaglio(){
    if(!bagagliaioVuoto()){
        bool estratto = false;
        for(int indice = 0; ((indice<DIMMAX)&&(!estratto)); indice=indice+1){
            if(bagagli[indice].presente){
                estratto = true;
                bagagli[indice].presente = false;
                num_bagagli--;
                return bagagli[indice].oggetto;
            }
        }
    }
}

int Bagagliaio::cercaBagaglio(Oggetto ogg){
    bool trovato = false;
    int indice;
    for(indice = 0; ((indice<DIMMAX) && (!trovato)); indice=indice+1){
        if(bagagli[indice].presente){
            if(bagagli[indice].oggetto.get_codice()==ogg.get_codice())
                trovato = true;
        }
    }
    if(trovato)
        return indice-1;
    else
        return -DIMMAX;
}

void Bagagliaio::vediBagagli(){
    if(!bagagliaioVuoto()){
        cout << num_bagagli << " scomparto/i occupato/i su " << DIMMAX << endl;
        for(int indice = 0; indice<DIMMAX; indice=indice+1){
            if(bagagli[indice].presente)
                cout << "Lo scomparto " << indice+1 << " contiene: " <<
                    bagagli[indice].oggetto.get_nome() << endl;
        }
    }
    else
        cout << "Il bagagliaio e' vuoto" << endl;
}

```

```

int Bagagliaio::cercaScompartoLibero(){
    if(!bagagliaioPieno()){
        bool trovato = false;
        int indice;
        for(indice = 0; ((indice<DIMMAX) && (!trovato)); indice=indice+1){
            if(!bagagli[indice].presente)
                trovato = true;
        }
        return indice-1;
    }
    else
        return -DIMMAX;
}

```

5.13.17 Autobus.h

```

#ifndef AUTOBUS_H
#define AUTOBUS_H

#include "Veicolo.h"

/* DEFINIZIONE DELLA CLASSE AUTOBUS
 * RISTRUTTURAZIONE DI Luca Carlucci
 */
using namespace std;

class Autobus: public Veicolo {

public:
    Autobus();
    //INIZIO MODIFICHE Luca Carlucci
    Autobus(string, int, int, unsigned short, bool);
    ~Autobus();
    bool autobusPieno();
private:
    bool postiLiberi;
    //FINE MODIFICHE Luca Carlucci
};

#endif /* AUTOBUS_H */

```

5.13.18 Autobus.cpp

```

#include "Autobus.h"

using namespace std;

```

```

//INIZIO MODIFICHE Luca Carlucci
Autobus::Autobus():Veicolo(){
    postiLiberi = false;
}

Autobus::Autobus(string n, int c, int l, unsigned short numP, bool inU):Veicolo(n, c, l, numP, inU){
    //Veicolo::set_indice_di_consumo(10);
    //Veicolo::set_livello_benzina(100);
    postiLiberi = false;
}

Autobus::~Autobus(){

}

bool Autobus::autobusPieno(){
    return(postiLiberi);
}
//FINE MODIFICHE Luca Carlucci

```

5.13.19 Automobile.h

```

/* HEADER FILE DI DEFINIZIONE DELLA CLASSE AUTOMOBILE
 * IL TIPO AUTOMOBILE E' UN PARTICOLARE TIPO DI
 * VEICOLO.
 */

#ifndef AUTOMOBILE_H_
#define AUTOMOBILE_H_

#include "Veicolo.h"
#include "Bagagliaio.h"

class Automobile: public Veicolo{
public:
    Automobile();
    Automobile(string, int, int, unsigned short, bool);
    ~Automobile();
    Bagagliaio bagagliaio;           //Un oggetto di tipo AUTOMOBILE dispone di un
    BAGAGLIAIO
};

#endif /* AUTOMOBILE_H_ */

```

5.13.20 Automobile.cpp

```

/* IMPLEMENTAZIONE DEGLI OPERATORI RELATIVI AL
 * TIPO AUTOMOBILE
 */

#include "Automobile.h"

Automobile::Automobile(){}

```

```

    }

Automobile::Automobile(string n, int c, int l, unsigned short numP, bool inU):Veicolo(n, c, l, numP, inU){

}

Automobile::~Automobile(){

}

```

5.14 MODIFICA CODICI LUOGHI – PALAGIANO MARCELLO

5.14.1 Modifica file “Mappa.nav”

Il codice nel progetto base1906 era il seguente:

```

36 34,Luogotrasporto,000000000500,via 8,1,1
37 40,Nella sala del teletrasporto dell'astronave aliena, 000000000000, via
      9,1,1
38 45,Roma,474600000000,via 6,8,8
39 46,Bari,450000130000,via 6,8,8
40 47,Pisa,004500000000,via 6,8,8

```

Il nuovo codice è:

```

36 34,Nella Palestra,003300000000,via 2,1,1
37 35,Nella sala giochi,000000150000, via 6,2,2
38 36,Nella biblioteca,000000000028, via 1,2,2
39 37,Nella vetrina,000000000028, via 1,2,2
40 38,Roma,403900000000,via 6,8,8
41 39,Bari,380000130000,via 6,8,8
42 40,Pisa,003800000000,via 6,8,8
43 41,Nel Meccanico,130000000000,via 1,2,2
44 42,Luogotrasporto,000000000500,via 8,1,1
45 48,Nella sala del teletrasporto dell'astronave aliena, 000000000000, via
      9,1,1

```

Il numero di luoghi è stato cambiato portandoli da 38 a 42.

Sono state modificate le righe di navigazione in modo che i luoghi possano essere raggiunti.

Il codice nel progetto base1906 era il seguente:

```

13,Nella Stazione di Servizio,00004600004,via 1,8,8
15,In una banca,230000140000,via 6,2,2

```

28,Nella Scuola,300129310000,via 1,1,1

33,Nella Stazione,001200000000,via 2,1,1

5,34,via 8,scendi,1,1,1,1

Il nuovo codice è:

13,Nella Stazione di Servizio,000039000004,via 1,8,8

15,In una banca,230035140000,via 6,2,2

28,Nella Scuola,300129313600,via 1,1,1

33,Nella Stazione,341200000000,via 2,1,1

5,42,via 8,scendi,1,1,1,1

33,34,via 2,nord,2,2,1,1

34,33,via 2,sud,2,2,1,1

35,15 via 6,ovest,2,2,1,1

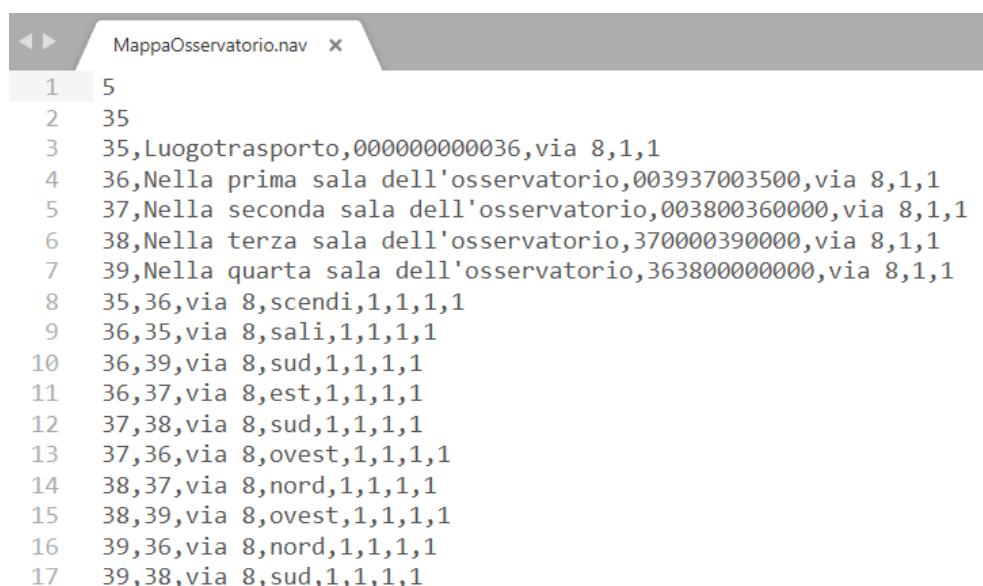
15,35 via 6,est,2,2,1,1

28,36,via 2,sali,2,2,1,1

36,28,via 2,scendi,2,2,1,1

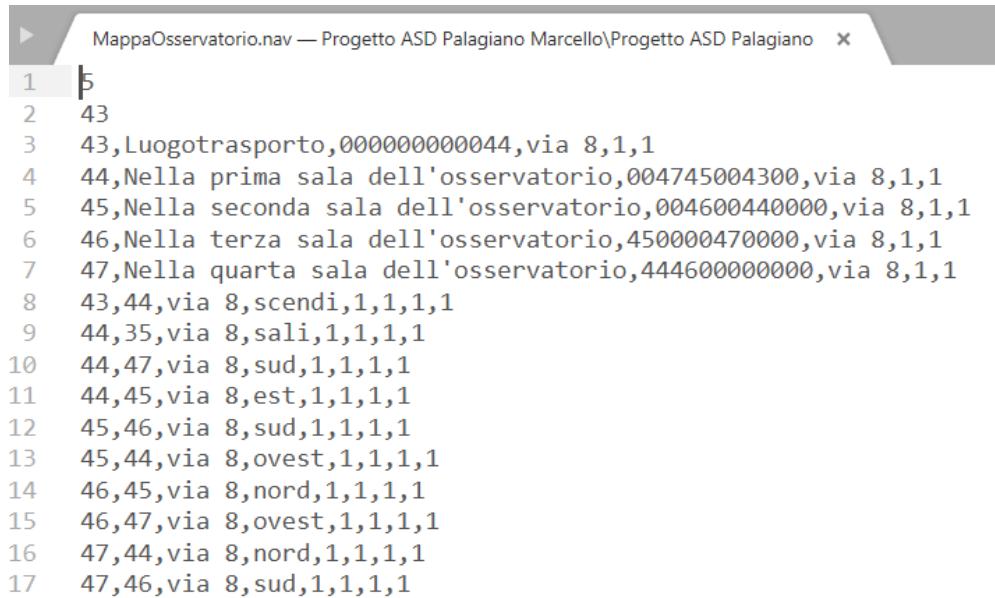
5.14.2 Modifica file “MappaOsservatorio.nav”

Il codice nel progetto base1906 era il seguente:



```
MappaOsservatorio.nav
1 5
2 35
3 35,Luogotrasporto,00000000036,via 8,1,1
4 36,Nella prima sala dell'osservatorio,003937003500,via 8,1,1
5 37,Nella seconda sala dell'osservatorio,003800360000,via 8,1,1
6 38,Nella terza sala dell'osservatorio,370000390000,via 8,1,1
7 39,Nella quarta sala dell'osservatorio,363800000000,via 8,1,1
8 35,36,via 8,scendi,1,1,1,1
9 36,35,via 8,sali,1,1,1,1
10 36,39,via 8,sud,1,1,1,1
11 36,37,via 8,est,1,1,1,1
12 37,38,via 8,sud,1,1,1,1
13 37,36,via 8,ovest,1,1,1,1
14 38,37,via 8,nord,1,1,1,1
15 38,39,via 8,ovest,1,1,1,1
16 39,36,via 8,nord,1,1,1,1
17 39,38,via 8,sud,1,1,1,1
```

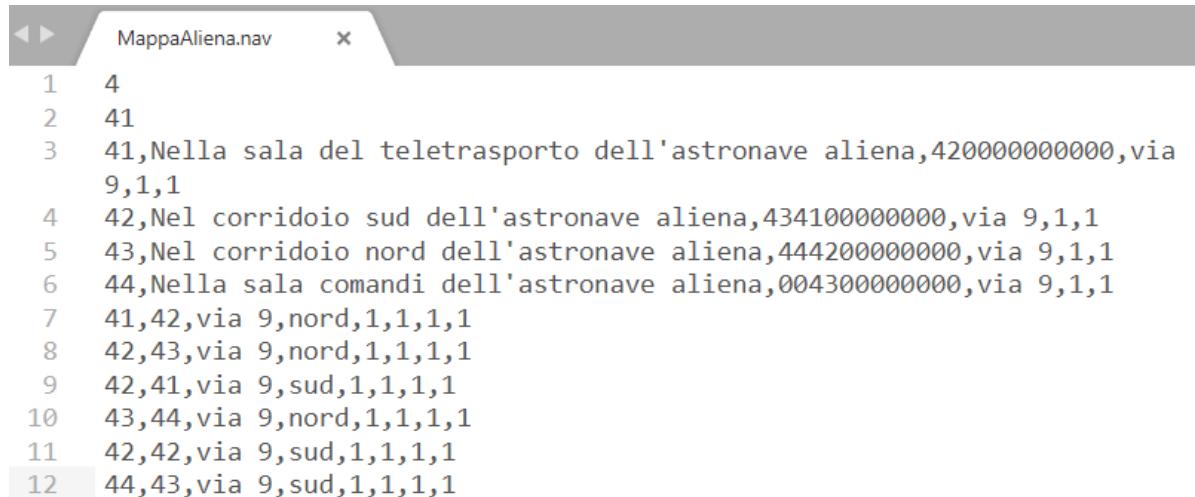
Il nuovo codice è:



```
MappaOsservatorio.nav — Progetto ASD Palagiano Marcello\Progetto ASD Palagiano ×  
1 §  
2 43  
3 43,Luogotrasporto,00000000044,via 8,1,1  
4 44,Nella prima sala dell'osservatorio,004745004300,via 8,1,1  
5 45,Nella seconda sala dell'osservatorio,004600440000,via 8,1,1  
6 46,Nella terza sala dell'osservatorio,450000470000,via 8,1,1  
7 47,Nella quarta sala dell'osservatorio,444600000000,via 8,1,1  
8 43,44,via 8,scendi,1,1,1,1  
9 44,35,via 8,sali,1,1,1,1  
10 44,47,via 8,sud,1,1,1,1  
11 44,45,via 8,est,1,1,1,1  
12 45,46,via 8,sud,1,1,1,1  
13 45,44,via 8,ovest,1,1,1,1  
14 46,45,via 8,nord,1,1,1,1  
15 46,47,via 8,ovest,1,1,1,1  
16 47,44,via 8,nord,1,1,1,1  
17 47,46,via 8,sud,1,1,1,1
```

5.14.3 Modifica file “MappaAliena.nav”

Il codice nel progetto base1906 era il seguente:



```
MappaAliena.nav ×  
1 4  
2 41  
3 41,Nella sala del teletrasporto dell'astronave aliena,420000000000,via  
9,1,1  
4 42,Nel corridoio sud dell'astronave aliena,434100000000,via 9,1,1  
5 43,Nel corridoio nord dell'astronave aliena,444200000000,via 9,1,1  
6 44,Nella sala comandi dell'astronave aliena,004300000000,via 9,1,1  
7 41,42,via 9,nord,1,1,1,1  
8 42,43,via 9,nord,1,1,1,1  
9 42,41,via 9,sud,1,1,1,1  
10 43,44,via 9,nord,1,1,1,1  
11 42,42,via 9,sud,1,1,1,1  
12 44,43,via 9,sud,1,1,1,1
```

Il nuovo codice è:

```
MappaAliena.nav — Progetto ASD Palagiano Marcello\Progetto ASD Palagiano x
1 4
2 49
3 49,Nella sala del teletrasporto dell'astronave aliena,500000000000,via 9,1,1
4 50,Nel corridoio sud dell'astronave aliena,514900000000,via 9,1,1
5 51,Nel corridoio nord dell'astronave aliena,525000000000,via 9,1,1
6 52,Nella sala comandi dell'astronave aliena,005100000000,via 9,1,1
7 49,50, via 9,nord,1,1,1,1
8 50,51, via 9,nord,1,1,1,1
9 50,49, via 9,sud,1,1,1,1
10 51,52, via 9,nord,1,1,1,1
11 50,50, via 9,sud,1,1,1,1
12 52,51, via 9,sud,1,1,1,1
```

5.14.4 Modifica file “Trasporti.nav”

Il codice nel progetto base1906 era il seguente:

```
Trasporti.nav — Progetto Algoritmi Marcello Palagiano\...Sorgente Progetto x
1 35,5,Mappa.nav
2 34,36,MappaOsservatorio.nav
3 41,11,Mappa.nav
4 11,41,MappaAliena.nav
```

Il nuovo codice è:

```
Trasporti.nav — Progetto ASD Palagiano Marcello\Progetto ASD Palagiano x
1 43,5,Mappa.nav
2 42,44,MappaOsservatorio.nav
3 49,11,Mappa.nav
4 11,49,MappaAliena.nav
r
```

5.14.5 Modifica righe di codice

Sono state modificate le righe di codice relative ai suddetti luoghi per i quali si è dovuto modificare il codice luogo.

azioni.inserisci(4900290048, 118); //Se si usa il teletrasporto nell'astronave aliena -- Palagiano Marcello:
modifica codice luogo da 40 a 48

```
//INIZIO modifiche DAVIDE MANTELLINI -- Palagiano Marcello: modifica codice luogo da 35 a 34
azioni.inserisci(3400290057, 120); //usa panca
azioni.inserisci(3400290047, 121); //usa tapis
azioni.inserisci(3400290099, 122); //usa terminale
azioni.inserisci(3400100099, 122); //guarda terminale
azioni.inserisci(3400290084, 123); //usa schede
azioni.inserisci(3400290097, 124); //usa macchinetta
//FINE modifiche DAVIDE MANTELLINI
```

```
//INIZIO modifiche GIACOMO CICALA -- Palagiano Marcello: modifica codice luogo da 36 a 35
azioni.inserisci(3500290086, 125); //usa slotmachine
//FINE modifiche GIACOMO CICALA
```

```
//inizio modifiche biblioteca Scatigna -- Palagiano Marcello: modifica codice luogo da 37 a 36 (biblioteca), da 38 a 37 (vetrina)
```

```
azioni.inserisci(3600250070, 126); //nuova azione etichetta su biblioteca
azioni.inserisci(3600220019, 127); //nuova azione di apertura su vetrina
azioni.inserisci(3700930019, 128); //nuova azione di chiusura su vetrina
azioni.inserisci(250090, 129); //leggi libro astronave
azioni.inserisci(250091, 129); //leggi libro equipaggio
//fine modifica biblioteca Scatigna
```

```
oggetti.inserisci(Oggetto("un teletrasporto",48,-49)); //-- Palagiano Marcello: modifica codice luogo da 41 a 49
```

```
//INIZIO modifiche DAVIDE MANTELLINI -- Palagiano Marcello: modifica codice luogo da 35 a 34
oggetti.inserisci(Oggetto("una panca per gli addominali", 57, -34));
oggetti.inserisci(Oggetto("un tapis roulant", 47, -34));
oggetti.inserisci(Oggetto("un terminale informativo",99, -34));
oggetti.inserisci(Oggetto("delle schede di allenamento", 84, 34));
```

```

oggetti.inserisci(Oggetto("una macchinetta",97, -34));
//FINE modifiche DAVIDE MANTELLINI

//INIZIO modifiche GIACOMO CICALA
oggetti.inserisci(Oggetto("una slotmachine", 86,-35));
oggetti.inserisci(Oggetto("5 euro",90,-99)); //Luogo -99 in quanto non visibili
oggetti.inserisci(Oggetto("10 euro",91,-99));
oggetti.inserisci(Oggetto("20 euro",92,-99));
oggetti.inserisci(Oggetto("50 euro",93,-99));
oggetti.inserisci(Oggetto("100 euro",94,-99));
//Fine modifiche GIACOMO CICALA

//inizio modifiche biblioteca Scatigna -- Palagiano Marcello: modifica codice luogo da 37 a 36 (biblioteca), da
38 a 37 (vetrina)
oggetti.inserisci(Oggetto("etichetta", 70, -36));
oggetti.inserisci(Oggetto("vetrina",19, -36));
oggetti.inserisci(Oggetto("astronave", 90, 37));
oggetti.inserisci(Oggetto("equipaggio", 91, 37));
//fine modifica Scatigna

```

5.14.5.1 Modifica codice Biblioteca – Vetrina

//Inizio modifiche Biblioteca Scatigna -- Palagiano Marcello: modifica codice luogo da 37 a 36 (biblioteca), da
38 a 37 (vetrina)

```

void Astro::scadenza()
{
    Lista<int>::posizione p = libriInPrestito.primolista();

    //DECREMENTO AUTOMATICO
    for (int j = 0;(!libriInPrestito.finlista(p)) && (j < MAX_LIBRI)); p = libriInPrestito.succlista(p))
    {
        int oggetto = libriInPrestito.leggilista(p);
        //cout << "oggetto: " << oggetto << "\n";
    }
}

```

```

int libro = oggetto - 90;

//cout << "libro : " << libro << " \n";

int scadenza = --scadenze[libro];
//cout << scadenza << " \n";;

if (scadenza == 10)
{
    for (i = 1; i <= oggetti.get_n_oggetti(); i++)
    {
        if (oggetti.get_oggetto(i).get_codice() == oggetto)
        {
            cout << "AVVISO: Mancano 10min per la restituzione del libro "
            << "\n      Corri in Biblioteca!\n";
            break;
        }
    }
}

else if (scadenza == 0)
{
    scadenze[libro] = 100;
    for (i = 1; i <= oggetti.get_n_oggetti(); i++)
    {
        if (oggetti.get_oggetto(i).get_codice() == oggetto)
        {
            oggetti.set_luogo(i, 37);

            cout << "\nIl " << oggetti.get_oggetto(i).get_nome() << " e' tornato in Biblioteca. \n\n";
            cout << "\nA causa della mancata restituzione hai perso 10 minuti del tuo tempo.";
            break;
        }
    }
}

```

```

        }

        libriInPrestito.canclista(p);

        tempo = tempo - 10; //PENALITA

    }

    j++;

}

//Fine modifiche Biblioteca Scatigna

//inizio modifiche biblioteca Scatigna -- Palagiano Marcello: modifica codice luogo da 37 a 36 (biblioteca), da
38 a 37 (vetrina)

void Astro::azione_126()

{
    interfaccia.scrivi("\n ---- REGOLAMENTO -----");

    interfaccia.scrivi("\n\n1) E' possibile prendere un libro in "
                      "\nprestito e restituirlo entro e non oltre 3 ore");

    interfaccia.scrivi("\n2) La restituzione e' valida solo se i libri "
                      "\nvengono lasciati al posto originario.");

    interfaccia.scrivi("\n3) Si consiglia di non lasciare i libri sparsi"
                      "\nin biblioteca");

    interfaccia.scrivi("\n4) Si prega infine di fare silenzio!");

    interfaccia.scrivi("\n\nSaluti dal Bibliotecario... \n");
}

```

5.14.5.2 Modifica codice Teletrasporto

//FINE modifiche SALVATORE VESTITA -- Palagiano Marcello: modifica codice luogo da 41 a 49

```

void Astro::azione_118() // Azione relativa al teletrasporto verso e dall'astronave aliena, gestita rispetto al
luogo_attuale

{
    if (luogo_attuale == 11) //se si è sulla poppa e si utilizza la pietra canalizzatrice, si arriva all'astronave
    aliena
    {

```

```

interfaccia.scrivi("Ti senti leggero...");

interfaccia.scrivi("E, senza che neanche tu te ne accorga, stai fluttuando verso un oggetto non identificato.");

interfaccia.scrivi("Utilizzando la pietra canalizzatrice, sei stato teletrasportato verso un'astronave aliena.");

storia_gioco.insStoria(stringa_comando, "utilizzando la pietra canalizzatrice, sei stato teletrasportato verso un'astronave aliena.");

//luogo_attuale = 41;

}

else if (luogo_attuale == 49) //se si utilizza il teletrasporto nell'astronave aliena, si torna sull'astronave

{
    interfaccia.scrivi("Stai tornando sull'astronave, pronto a riprendere la tua avventura.");

    storia_gioco.insStoria(stringa_comando, "sei tornato sulla Neutronia azionando il teletrasporto della nave aliena.");

    //luogo_attuale = 11;

}

trasporto(luogo_attuale);

}

```

5.15 IMPLEMENTAZIONE MECCANICO

Per effettuare l'integrazione nel progetto base, i file che sono stati modificati e/o aggiunti sono i seguenti:

- Astro.h
- Astro.cpp
- Attivita.h
- Attivita.cpp
- Batteria.h
- Batteria.cpp
- Gioco.h
- Gioco.cpp
- Mappa.nav

Nella cartella “descrizioni”, inoltre, è stato aggiunto il seguente file contenente la descrizione del nuovo luogo inserito:

- 41.txt

5.15.1 Aggiunta del file “Batteria.h”

Batteria.h è la classe relativa agli oggetti “batteria”.

```
#ifndef BATTERIA_H
#define BATTERIA_H
#include <iostream>
#include <string>
using namespace std;
class Batteria{
public:
    Batteria();
    virtual ~Batteria();
    int get_stato();
    string get_modello();
    void set_stato(int stat);
    void set_modello(string mod);
private:
    int stato;
    string modello;
};
#endif // BATTERIA_H
```

5.15.2 Aggiunta del file “Batteria.cpp”

Batteria.cpp contiene l’implementazione dei metodi definiti in Batteria.h

```
#include "Batteria.h"
```

```
Batteria::Batteria(){}
Batteria::~Batteria(){}
int Batteria::get_stato(){
```

```
    return stato;
}
```

```

string Batteria::get_modello(){
    return modello;
}

void Batteria::set_stato(int stat){
    stato = stat;
}

void Batteria::set_modello(string mod){
    modello = mod;
}

```

5.15.3 Aggiunta del file “Attivita.h”

Attivita.h è la classe relativa agli oggetti “attivita”.

```

#ifndef ATTIVITA_H
#define ATTIVITA_H

#include <iostream>
#include <string>

using namespace std;
class Attivita{
public:
    Attivita();
    virtual ~Attivita();
    string get_data();
    string get_descrizione();
    string get_autore();
    void set_data(string d);
    void set_descrizione(string desc);
    void set_autore(string a);
private:
    string data;
}

```

```
    string descrizione;
    string autore;
};

#endif // ATTIVITA_H
```

5.15.4 Aggiunta del file “Attivita.cpp”

Attivita.cpp contiene l’implementazione dei metodi definiti in Attivita.h

```
#include "Attivita.h"

Attivita::Attivita(){
    //ctor
}

Attivita::~Attivita(){
    //dtor
}

string Attivita::get_data(){
    return data;
}

string Attivita::get_descrizione(){
    return descrizione;
}

string Attivita::get_autore(){
    return autore;
}

void Attivita::set_data(string d){
    data = d;
}

void Attivita::set_descrizione(string desc){
    descrizione = desc;
}

void Attivita::set_autore(string a){
    autore = a;
}
```

}

5.15.5 Modifica al file “Gioco.cpp”

Nel file Gioco.cpp sono state apportate delle modifiche al codice del progetto base, per permettere di raggiungere, sia con l’*automobile* che con l’*autobus*, il luogo “Meccanico”. Inoltre, è stato aggiunto il controllo che non permette di raggiungere la città di “Bari” con l’*automobile* se prima non viene fatta cambiare la batteria dal *meccanico*.

Infine, è stata aggiunto il controllo che impedisce di prendere oggetti se si è a bordo dell’*automobile* e dell’*autobus*.

```
//INIZIO modifiche PALAGIANO MARCELLO

if (((autobus.get_inUso()==true) || (automobile.get_inUso()==true)) && (a<38 && a!=13 && a!=41)){

    if(autobus.get_inUso()==true){

        interfaccia.scrivi("- Non puoi andare li' con l'autobus");
        stringa_risposta = "hai provato ad andare con l'autobus in posti non autorizzati";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }

    if(automobile.get_inUso()==true){

        interfaccia.scrivi("- Non puoi andare li' con l'auto");
        stringa_risposta = "hai provato ad andare con l'auto in posti non autorizzati";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }

    cambia_dir = false; //Modifica Rosita Galiandro
} else{
    if(automobile.get_inUso()==true && batteria_cambiata==false && a!=41 && a!=13){

        interfaccia.scrivi("Sarebbe pericoloso andare di la' con la batteria scarica!");
        stringa_risposta = "non hai cambiato la batteria";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }
}

else{
    int b = a - 9; //Aggiustamento per ricerca luogo da file DELLA FOLGORE GRAZIA
    if((a>=38) && (a<=40) && ((mappa.get_nome_luogo(b) == "Bari") || (mappa.get_nome_luogo(b) == "Roma") || (mappa.get_nome_luogo(b) == "Pisa"))){

        a = b;
    }
}
```

```

if((autobus.get_inUso()==false)&&(automobile.get_inUso()==false)){
    interfaccia.scrivi("- Impieghi piu' tempo muovendoti a piedi");
    tempo = tempo - 3;
}
luogo_attuale = a;
if(mappa.get_nome_luogo(a) != "Luogotrasporto"){
    stringa_risposta = "sei andato " + mappa.get_nome_luogo(luogo_attuale) + ".";
    //Modifica PMF(storia)
    storia_gioco.insStoria(stringa_comando , stringa_risposta);
    //Modifica PMF(storia)
}
//FINE modifiche PALAGIANO MARCELLO

//INIZIO modifiche PALAGIANO MARCELLO
else if(oggetti.get Oggetto(117).get_luogo() == 0){
    interfaccia.scrivi("** Per prendere oggetti devi lasciare l'autobus! **");
    cpz=false;
}
else if(oggetti.get Oggetto(119).get_luogo() == 0){
    interfaccia.scrivi("** Per prendere oggetti devi lasciare l'automobile! **");
    cpz=false;
}
//FINE modifiche PALAGIANO MARCELLO

```

5.15.6 Modifica al file “Gioco.h”

Nel file Gioco.h sono state apportate delle modifiche al codice, al fine di poter gestire la *pila di batterie*, il *diario delle attivita* e la *coda di attesa del meccanico* all'interno del gioco.

Di seguito sono riportate le righe di codice aggiunte al file:

```

//INIZIO modifica PALAGIANO MARCELLO
#include "Batteria.h" // Batterie che si trovano nel luogo Meccanico

```

```

#include "Attivita.h" // Contenuto del diario nel luogo Meccanico
//FINE modifica PALAGIANO MARCELLO

//INIZIO modifiche PALAGIANO MARCELLO

Batteria batteria;
Pila<Batteria> batterie;
Pila<Batteria> batterie_scariche;
Pila<Batteria> batterie_appoggio;
Attivita attivita;
Lista<Attivita> elenco;
Lista<Attivita>::posizione attivitaSelezionata;
Personaggi cliente;
Coda<Personaggi> coda_attesa;
Coda<Personaggi> codaa_appoggio;
bool batteria_cambiata;
//FINE modifiche PALAGIANO MARCELLO

```

5.15.7 Modifica al file “Astro.h”

In questo file sono state aggiunte alcune righe di codice come dichiarazione delle azioni implementate.

```

//INIZIO modifiche PALAGIANO MARCELLO

void azione_130(); //guarda banco da lavoro
void azione_131(); //guarda ricambio astronave
void azione_132(); //guarda pila di batterie
void azione_133(); //guarda batterie scariche
void azione_134(); //guarda chiave inglese
void azione_135(); //guarda diario
void azione_136(); //leggi diario
void azione_137(); //guarda cartello
void azione_138(); //guarda computer
void azione_139(); //parla meccanico
//FINE modifiche PALAGIANO MARCELLO

```

5.15.8 Modifica al file “Astro.cpp”

In questo file è stata modificata la procedura *init_specifiche()*, alla quale si sono aggiunti vocaboli, oggetti e azioni e la procedura *esegui_specifiche(int a, Mappa &M)*, in più sono state implementate le azioni da 130 a 139.

Inoltre, è stato modificato il codice del progetto base di Della Folgore per far interagire l’*automobile* con il *meccanico*.

5.15.8.1 Nuovi vocaboli

```
//INIZIO modifiche PALAGIANO MARCELLO
```

```
vocabolario.inserisci("banco", 43);  
vocabolario.inserisci("ricambio", 119);  
vocabolario.inserisci("pila", 70);  
vocabolario.inserisci("batterie", 16);  
vocabolario.inserisci("chiave_inglese", 118);  
vocabolario.inserisci("diario", 57);  
vocabolario.inserisci("meccanico", 73);  
//FINE modifiche PALAGIANO MARCELLO
```

5.15.8.2 Nuovi oggetti

```
//INIZIO modifiche PALAGIANO MARCELLO
```

```
oggetti.inserisci(Oggetto("un banco da lavoro", 43, -41));  
oggetti.inserisci(Oggetto("un pezzo di ricambio per astronave", 119, -41));  
oggetti.inserisci(Oggetto("una pila di batterie", 70, -41));  
oggetti.inserisci(Oggetto("una chiave_inglese", 118, 41));  
oggetti.inserisci(Oggetto("un diario", 57, -41));  
oggetti.inserisci(Oggetto("un cartello", 60, -41));  
oggetti.inserisci(Oggetto("un computer", 99, -41));  
oggetti.inserisci(Oggetto("un meccanico", 73, -41));  
//FINE modifiche PALAGIANO MARCELLO
```

5.15.8.3 Modifiche azioni

```
//INIZIO modifiche PALAGIANO MARCELLO
```

```
azioni.inserisci(4100100043, 130); //guarda banco da lavoro  
azioni.inserisci(4100100119, 131); //guarda ricambio astronave
```

```

azioni.inserisci(4100100070, 132); //guarda pila batterie
//133 definita se si scartano batterie dalla pila --- guarda batterie
azioni.inserisci(100118, 134); //guarda la chiave_inglese
azioni.inserisci(4100100057, 135); //guarda il diario
azioni.inserisci(4100250057, 136); //leggi il diario
azioni.inserisci(4100100060, 137); //guarda il cartello
azioni.inserisci(4100100099, 138); //guarda computer
azioni.inserisci(4100390073, 139); //parla meccanico
//FINE modifiche PALAGIANO MARCELLO

```

5.15.8.4 Inizializzazione oggetti stanza meccanico

Il pezzo di codice seguente riporta l'inizializzazione della variabile *batteria_cambiata* utilizzata per verificare se il *meccanico* ha sostituito o meno la batteria, il riempimento della *pila di batterie*, inserendo alcuni oggetti in essa, l'aggiunta di alcune *attività* all'interno del *diario* e l'inserimento di alcune persone all'interno della *coda di attesa* del meccanico.

```

//INIZIO modifiche PALAGIANO MARCELLO
batteria_cambiata=false;
batteria.set_modello("A0001");
batteria.set_stato(100);
batterie.inpila(batteria);
batteria.set_modello("A0002");
batteria.set_stato(35);
batterie.inpila(batteria);
batteria.set_modello("A0003");
batteria.set_stato(35);
batterie.inpila(batteria);
batteria.set_modello("A0004");
batteria.set_stato(55);
batterie.inpila(batteria);
attivitaSelezionata = elenco.primolista();
attivita.set_data("01 Gennaio 2003");
attivita.set_autore("Capo Officina");
attivita.set_descrizione("Eseguire collaudo su pezzo riparato nel motore");

```

```

elenco.inslista(attivita, attivitaSelezionata);
attivita.set_data("22 Marzo 2000");
attivita.set_autore("Capo Officina");
attivita.set_descrizione("Eseguita verifica impianto di raffreddamento motore");
elenco.inslista(attivita, attivitaSelezionata);
attivita.set_data("10 Giugno 2010");
attivita.set_autore("Capo Officina");
attivita.set_descrizione("Riparare ventola di aerazione");
elenco.inslista(attivita, attivitaSelezionata);
cliente.setNome("Frank");
coda_attesa.incoda(cliente);
cliente.setNome("John");
coda_attesa.incoda(cliente);
cliente.setNome("Kirk");
coda_attesa.incoda(cliente);
cliente.setNome("Michael");
coda_attesa.incoda(cliente);
//FINE modifiche PALAGIANO MARCELLO

```

5.15.8.5 Modifica esegui_specifiche

Nella procedura *esegui_specifiche(int a, Mappa &M)* sono state aggiunte e adattate le seguenti righe di codice:

```
//INIZIO modifiche PALAGIANO MARCELLO
```

```
case 130:
```

```
azione_130();
```

```
break;
```

```
case 131:
```

```
azione_131();
```

```
break;
```

```
case 132:
```

```
azione_132();
```

```

break;

case 133:
azione_133();
break;

case 134:
azione_134();
break;

case 135:
azione_135();
break;

case 136:
azione_136();
break;

case 137:
azione_137();
break;

case 138:
azione_138();
break;

case 139:
azione_139();
break;

//FINE modifiche PALAGIANO MARCELLO

```

5.15.8.6 Implementazione procedure per le azioni 130-139

5.15.8.6.1 Azione 130 Guarda banco da lavoro

//INIZIO modifiche PALAGIANO MARCELLO

```

void Astro::azione_130() //guarda banco da lavoro
{
    interfaccia.scrivi("vedo un banco da lavoro con alcuni appunti e progetti");
}

```

5.15.8.6.2 Azione 131 Guarda ricambio

```
void Astro::azione_131() //esamina ricambio astronave
{
    interfaccia.scrivi("Vedo un pezzo di ricambio per l'astronave... Purtroppo e' rotto!");
}
```

5.15.8.6.3 Azione 132 Guarda pila

```
void Astro::azione_132() //guarda pila di batterie
{
    if (oggetti.get_oggetto(117).get_luogo() == 0){
        interfaccia.scrivi("Devi prima lasciare l'autobus");
        stringa_risposta = "Ti e' stato detto di lasciare prima l'autobus.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }
    else if(oggetti.get_oggetto(119).get_luogo() == 0){
        interfaccia.scrivi("Devi prima lasciare l'automobile");
        stringa_risposta = "Ti e' stato detto di lasciare prima l'automobile.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }
    else{
        string risposta;
        while(!batterie.pilavuota()){
            batteria = batterie.leggipila();
            cout<<"La batteria "<<batteria.get_modello()<<" ha una carica del "<<batteria.get_stato()
            <<"%"<<"\n\n";
            if(batteria.get_stato() < 50){
                cout<<"La carica della batteria "<<batteria.get_modello()<<" non e' sufficiente."<<" Vuoi
                scartare la batteria? ";
                cin>>risposta;
                cout<<"\n";
                if((risposta=="s") || (risposta=="si")){
                    batterie_scariche.inpila(batteria);
                    batterie.fuoripila();
                }
            }
        }
    }
}
```

```

immetti += 1;

if (immetti == 1){

oggetti.inserisci(Oggetto("delle batterie scariche", 16, -41));
azioni.inserisci(4100100016, 133); //guarda batterie scariche

}

}

else{

batterie_appoggio.inpila(batteria);

batterie.fuoripila();

}

}

else{

batterie_appoggio.inpila(batteria);

batterie.fuoripila();

}

}

//Ripristina la pila originaria, con le batterie che non sono state scartate

while(!batterie_appoggio.pilavuota()){

    batteria = batterie_appoggio.leggipila();

    batterie.inpila(batteria);

    batterie_appoggio.fuoripila();

}

}

}

```

5.15.8.6.4 Azione 133 Guarda batterie

```

void Astro::azione_133() //guarda batterie scariche

{

cout << "Qui ci sono le batterie scariche che hai scartato dalla pila!\n";

while(!batterie_scariche.pilavuota()){

    cout << "\n- Modello: " << batterie_scariche.leggipila().get_modello() << " - Carica: " <<
batterie_scariche.leggipila().get_stato() << "%";

```

```

batterie_appoggio.inpila(batterie_scariche.leggipila());
batterie_scariche.fuoripila();
}

cout << "\n\n";

while(!batterie_appoggio.pilavuota()) //ripristino pila batterie originale{
    batterie_scariche.inpila(batterie_appoggio.leggipila());
    batterie_appoggio.fuoripila();
}

}

```

5.15.8.6.5 Azione 134 Guarda chiave_inglese

```

void Astro::azione_134() //guarda chiave_inglese

{
    interfaccia.scrivi("Questa chiave ha una forma molto particolare.\nAprira' qualcosa? O sara' solo un
attrezzo meccanico?\n");
}

}

```

5.15.8.6.6 Azione 135 Guarda diario

```

void Astro::azione_135() //guarda diario

{
    interfaccia.scrivi("Sembra un diario delle attivita'. Magari potrebbe essere utile leggerlo!");
}

}

```

5.15.8.6.7 Azione 136 Leggi diario

```

void Astro::azione_136() //leggi diario

{
if (oggetti.get Oggetto(117).get_luogo() == 0){

    interfaccia.scrivi("Devi prima lasciare l'autobus");

    stringa_risposta = "Ti e' stato detto di lasciare prima l'autobus.";

    storia_gioco.insStoria(stringa_comando , stringa_risposta);

}

else if(oggetti.get Oggetto(119).get_luogo() == 0){

    interfaccia.scrivi("Devi prima lasciare l'automobile");
}
}

```

```

        stringa_risposta = "Ti e' stato detto di lasciare prima l'automobile.";

        storia_gioco.insStoria(stringa_comando , stringa_risposta);

    }

    else{

bool continua = true;

string ris;

attivitaSelezionata = elenco.primolista();

while( (!elenco.finlista(attivitaSelezionata)) && (continua)) {

cout<<"Autore: " <<elenco.leggilista(attivitaSelezionata).get_autore() <<"\n";
cout<<"\t" <<elenco.leggilista(attivitaSelezionata).get_descrizione() <<"\n";
cout<<"Data: " <<elenco.leggilista(attivitaSelezionata).get_data() <<"\n\n";
cout<<"Vuoi andare AVANTI o INDIETRO? ";

cin>>ris;

cout<<"\n\n";

if((ris=="avanti") || (ris == "avant") || (ris== "avan") || (ris == "ava") || (ris == "av") || (ris == "a")){
    attivitaSelezionata = elenco.succlista(attivitaSelezionata);
    if(elenco.finlista(attivitaSelezionata))
    {
        cout << "Diario Finito!\n";
    }
}

else if ((ris=="indietro") || (ris == "indietr") || (ris== "indiet") || (ris == "indie") || (ris == "indi") || (ris == "ind") || (ris == "in") || (ris == "i")){
    if(attivitaSelezionata != elenco.primolista()){
        attivitaSelezionata = elenco.prelista(attivitaSelezionata);
    }
}

else{
    continua = false;
    cout<<"Ho chiuso il diario\n\n";
}

}

else{

```

```

cout<<"Ho chiuso il diario\n\n";
continua = false;
}
}
}

```

5.15.8.6.8 Azione 137 Guarda cartello

```

void Astro::azione_137() //guarda cartello
{
cout<<"*****\n";
cout<<"* OFFICINA. Per qualsiasi richiesta rivolgersi al personale autorizzato! *\n";
cout<<"*****\n";
}
```

5.15.8.6.9 Azione 138 Guarda computer

```

void Astro::azione_138() //guarda computer
{
    if (oggetti.get Oggetto(117).get_Luogo() == 0){
        interfaccia.scrivi("Devi prima lasciare l'autobus");
        stringa_risposta = "Ti e' stato detto di lasciare prima l'autobus.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }
    else if(oggetti.get Oggetto(119).get_Luogo() == 0){
        interfaccia.scrivi("Devi prima lasciare l'automobile");
        stringa_risposta = "Ti e' stato detto di lasciare prima l'automobile.";
        storia_gioco.insStoria(stringa_comando , stringa_risposta);
    }
    else{
        int r, i = 5, num=0;
        string nome;
        do{
            cout<<"\nIl computer e' acceso. Cosa vuoi fare?\n";

```

```

cout<<"(1) Gestisci prenotazioni\n";
cout<<"(2) Visualizza persone in attesa\n";
cout<<"(3) Prenotati\n";
cout<<"(4) Spegni PC\n\n";
cin>>r;
cout<<"\n";
if (!cin){
    cin.clear();
    cin.ignore(256, '\n');
    //system("clear"); //linux os
    interfaccia.scrivi("Input errato");
    fflush(stdin);
}
else{
    switch(r){
        case 1:
            cout<<"Non puoi accedere a questa sezione! Le prenotazioni possono essere gestite solo dal capo officina.\n";
            break;
        case 2:
            cout<<"In ordine di arrivo, ci sono le seguenti persone in attesa:\n";
            i = 1;
            while(!coda_attesa.codavuota()){
                cout<<i << ". " <<coda_attesa.leggicoda().getNome() <<"\n";
                codaa_appoggio.incoda(coda_attesa.leggicoda());
                coda_attesa.fuoricoda();
                i++;
            }
            while(!codaa_appoggio.codavuota()){
                coda_attesa.incoda(codaa_appoggio.leggicoda());
                codaa_appoggio.fuoricoda();
            };
    };
}

```

```

break;

case 3:

cout<<"Inserisci il tuo nome: ";
cin>>nome;
cliente.setNome(nome);
coda_attesa.incoda(cliente);
num=0;

while(!coda_attesa.codavuota()){

codaa_appoggio.incoda(coda_attesa.leggicoda());
coda_attesa.fuoricoda();

num++;
}

while(!codaa_appoggio.codavuota()){

coda_attesa.incoda(codaa_appoggio.leggicoda());
codaa_appoggio.fuoricoda();
};

cout<<"\nPrenotazione effettuata.\nSei il numero " <<num <<"\n";
break;

case 4:

cout<<"Spengo il PC.\n\n";
break;

}

}

}while(r != 4);

}
}
```

5.15.8.6.10Azione 139 Parla meccanico
void Astro::azione_139() //parla meccanico

```
{  
    if(!batteria_cambiata)  
    {  
        interfaccia.scrivi("Salve! Sono il meccanico");  
    }  
}
```

```

        if (oggetti.get Oggetto(119).get Luogo() != 41 && oggetti.get Oggetto(119).get Luogo() != 0){ //se
non c'è l'automobile

    interfaccia.scrivi("Per cambiare la batteria dell'automobile devi portarla qui.");
    stringa_risposta = "ti e' stato detto di venire con l'automobile.";
    storia_gioco.insStoria(stringa_comando , stringa_risposta);

}

else{

    string risp;
    do{
        risp = interfaccia.leggi_stringa("Vuoi cambiare la batteria
dell'automobile? Ci vorra' del tempo pero' [Si/No]");
    }

    while(risp != "Si" && risp != "No" && risp != "s" && risp != "n" &&
risp != "si" && risp != "no");

    if(risp == "Si" || risp == "s" || risp == "si")
    {

        string risposta;
        batteria_cambiata = true; //La batteria è stata cambiata

        interfaccia.scrivi("Batteria cambiata! Ci vediamo!");
        tempo = tempo - 20;
        interfaccia.scrivi("**** Il tuo tempo e' stato decrementato di 20 punti ****");
    }

    else
        interfaccia.scrivi("Ok, sono qui se hai bisogno!");

}

}

else
    interfaccia.scrivi("Hai gia' cambiato la batteria dell'automobile!");

}

//FINE integrazione PALAGIANO MARCELLO

```

5.15.8.7 Modifica codice progetto base

È stato inserito un controllo per verificare se la batteria è stata cambiata dal *meccanico*:

```
//INIZIO modifiche PALAGIANO MARCELLO  
if(batteria_cambiata==false){  
interfaccia.scrivi("Accidenti! La batteria e' quasi scarica! Meglio andare da un meccanico.");  
}  
//FINE modifiche PALAGIANO MARCELLO
```

È stato inserito un controllo per verificare dove è possibile lasciare l'*automobile* e l'*autobus*:

```
//INIZIO modifiche PALAGIANO MARCELLO  
if (luogo_attuale == 13 || (mappa.get_nome_luogo(luogo_attuale) == "Bari") ||  
(mappa.get_nome_luogo(luogo_attuale) == "Roma") || (mappa.get_nome_luogo(luogo_attuale) == "Pisa")  
|| luogo_attuale==41){  
//FINE modifiche PALAGIANO MARCELLO
```

5.15.9 Modifica al file “Mappa.nav”

Per poter aggiungere il luogo Meccanico si è dovuto modificare il file Mappa.nav, cambiando in esso il numero dei luoghi, portandoli da 42 a 43, e aggiornando le righe di navigazione in modo che il luogo inserito possa essere raggiunto.

Il luogo Meccanico è stato inserito a sud rispetto alla Stazione di servizio.

13,Nella Stazione di Servizio,004139000004,via 1,8,8

41,Nel Meccanico,130000000000,via 1,2,2

13,41,via 1,sud,1,1,1,1

41,13,via 1,nord,1,1,1,1

5.16 CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA”

5.16.1 Verbo “ruba”

```
void Gioco::ruba()      //azione RUBA
{
    if (oggetti.get_oggetto(og).get_luogo() == 0 && oggetti.get_oggetto(og).get_rubato() == true){
        interfaccia.scrivi(" L'hai gia' rubato.");
    }
    else if (oggetti.get_oggetto(og).get_luogo() < 0){
        interfaccia.scrivi(" Non puoi rubarlo.");
    }
    else {
        if (og == 49 || og == 118 || og == 73){      //puo rubare: motorino, chiave_auto e buono pasto
            int l = 0;
            oggetti.set_luogo(og, l);
            oggetti.set_rubato(og,true);
            interfaccia.scrivi("Rubato.");
        }
        else {
            interfaccia.scrivi ("Quest'oggetto non puo essere rubato. Per averlo devi prenderlo.");
        }
    }
    interfaccia.a_capo();
}
```

5.16.2 Verbo “rubati”

```
void Gioco::rubati()      //Inventario oggetti rubati
{
    interfaccia.elenca_rubati(oggetti.posizionati_in(0),"Hai rubato:\n");
}
```

5.16.3 Personaggio“carabiniere”

```
//inizio modifiche ROSA CHIARAPPA
p101.setNome("carabiniere");
p101.setFrasi("Il carabiniere ti ha perquisito e ha trovato in tuo possesso un oggetto rubato");
p101.setLuogo(2);
insPersonaggi->inserisci(94, p101);
//fine modifiche ROSA CHIARAPPA

//inizio modifiche ROSA CHIARAPPA

void Gioco::aggiorna_luogo_carabiniere(int contatore){

    int codice = 94;
    Personaggi carabiniere = insPersonaggi.recupera(codice);  //
    if (contatore >= 10){
        srand (time(NULL));
        carabiniere.setLuogo(rand() % 48 + 1);
        insPersonaggi.aggiorna(codice, carabiniere);
    }
// std::cout <<"Il carabiniere e' nella stanza: "<< carabiniere
}
```

```

bool Gioco::verifica_presenza_carabiniere(int luogo_attuale){

    int codice = 94;
    Personaggi carabiniere = insPersonaggi.recupera(codice);
    if (carabiniere.getLuogo() == luogo_attuale){
        return true;
    } else {
        return false;
    }
}

void Gioco::perquisizione_giocatore(int &luogo_attuale){

    if (oggetti.get Oggetto(118).get_rubato() == true || oggetti.get Oggetto(73).get_rubato() == true || oggetti.get Oggetto(49).get_rubato() == true)
    {
        interfaccia.scrivi("Il carabiniere ti ha perquisito e ha trovato in tuo possesso un oggetto rubato, sei in arresto!");
        interfaccia.a_capo();
        luogo_attuale = 53-9;
        interfaccia.scrivi("Puoi uscire dalla caserma fra 30 secondi.");
        sleep (30);
        interfaccia.scrivi("Ora sei libero.");
    }
    else
    {
        interfaccia.scrivi("Il carabiniere ti ha perquisito e non ha trovato nessun oggetto rubato in tuo possesso.");
    }
    interfaccia.a_capo();
}

```

5.16.4 Luogo "caserma"

53,Nella caserma,030000000000,via 1,1,1

5.17 ZAINO E VALIGIA

5.17.1 Astro.h

Sono state aggiunte le seguenti funzioni:

```

// Modifiche_ML
bool indossa_specifiche();
bool indosso_specifiche();

```

5.17.2 Astro.cpp

Sono stati aggiunti i seguenti vocaboli:

```

vocabolario.inserisci("valigia", 155);
vocabolario.inserisci("zaino", 156);
vocabolario.inserisci("raccogli", 258);
vocabolario.inserisci("aiuto", 259);

```

```
vocabolario.inserisci("help", 259);
```

Sono stati aggiunte le seguenti azioni:

```
// Modifiche Matteo_Luceri(ex_Sternativo)

azioni.inserisci(80155, -2); //Azione prendi valigia
azioni.inserisci(80156, -2); //Azione prendi zaino
azioni.inserisci(90155, 3); //Azione lascia valigia
azioni.inserisci(90156, 3); //Azione lascia zaino
azioni.inserisci(100155, 140); //Azione guarda valigia
azioni.inserisci(100156, 141); //Azione guarda zaino
azioni.inserisci(1550000, 143); //Azione inventarla valigia
azioni.inserisci(1560000, 144); //Azione inventario zaino
azioni.inserisci(2590000, 145); //Azione aiuto/help
```

Sono state modificate le seguenti azioni:

```
azioni.inserisci(200050, -142); //Modifica ML
azioni.inserisci(200051, -142); //Modifica ML
azioni.inserisci(200052, -142); //Modifica ML
```

In tutti gli oggetti è stato aggiunto un attributo "peso" e sono stati aggiunti:

```
//_Modifiche_Matteo_Luceri
oggetti.inserisci(Oggetto("una valigia", 155, 6, 0));
oggetti.inserisci(Oggetto("uno zaino", 156, 6, 0));
```

Sono state inizializzate le seguenti variabili:

```
//Modifiche ML
aperto=false;
n_oggettiZ = 0;
peso_MaxZ = 5;
n_oggettiZV = 0;
peso_MaxZV = 8;
e11 = 0;
e12 = 0;
e13 = 0;
e14 = 0;
e15 = 0;
e16 = 0;
//fine modifiche
```

Sono state modificate le seguenti funzioni:

Le funzioni preso_specifiche e prendi_specifiche sono state "spostate" (con le dovute modifiche elencate di seguito) in indosso_specifiche e indossa_specifiche e le prime due sono state modificate in questo modo:

```
if(og == 143 && oggetti.get_oggetto(144).get_luogo() == 0){
    // controllo che l'oggetto sia la valigia e che nell'inventario ci sia lo zaino
    interfaccia.scrivi("Lascia prima lo zaino.");
}
else if(og == 144 && oggetti.get_oggetto(143).get_luogo() == 0){
    // controllo che l'oggetto sia lo zaino e che nell'inventario ci sia la valigia
    interfaccia.scrivi("Lascia prima la valigia.");
}
```

Sono state aggiunte le seguenti azioni:

```
void Astro::azione_140(){           //Azione guarda valigia
    interfaccia.scrivi("E' la valigia del secondo pilota.");
    interfaccia.scrivi("Sembra molto capiente.");
}
void Astro::azione_141(){           //Azione guarda zaino
    interfaccia.scrivi("E' il tuo zaino.");
    interfaccia.scrivi("Puo' esserti utile per trasportare oggetti.");
}
```

```

bool Astro::preso_specifiche() {
    //Modifiche Mirco Sternativo
    bool avvisato = true;

    if ((og == 11 || og == 4 || og == 23 || og == 25) && (oggetti.get_oggetto(143).get_luogo() == 0)){
        if(oggetti.get_oggetto(og).get_peso() > peso_MaxZV && og != 25){
            // controllo che il peso dell'oggetto sia maggiore allo spazio disponibile
            // nella valigia e che l'oggetto sia diverso dal manuale
            oggetti.set_luogo(og,luogo_attuale);
            interfaccia.scrivi("La valigia e' troppo piena..."); 
            interfaccia.scrivi("(Suggerimento: togli qualcosa dalla valigia.)");
        } else if(oggetti.get_oggetto(og).get_peso() > peso_MaxZV && og == 25){
            oggetti.set_luogo(og,0);
            interfaccia.scrivi("La valigia e' troppo piena..."); 
            interfaccia.scrivi("Lo tengo in mano, e' troppo importante."); 
            interfaccia.scrivi("(Suggerimento: controlla nell'inventario)");
        } else{
            interfaccia.scrivi("Ora e' in valigia.");
            ins.inserisci(oggetti.get_oggetto(og).get_codice());
            n_oggettiZV++;
            peso_MaxZV -= oggetti.get_oggetto(og).get_peso();
        } else if((og == 11 || og == 4 || og == 23 || og == 25) &&
                  (oggetti.get_oggetto(144).get_luogo() == 0)){
            if(oggetti.get_oggetto(og).get_peso() > peso_MaxZ && og != 25){
                // controllo che il peso dell'oggetto sia maggiore allo spazio
                // disponibile nella valigia e che l'oggetto sia diverso dal manuale
                oggetti.set_luogo(og,luogo_attuale);
                interfaccia.scrivi("Lo zaino e' troppo pieno..."); 
                interfaccia.scrivi("(Suggerimento: togli qualcosa dallo zaino.)");
            } else if(oggetti.get_oggetto(og).get_peso() > peso_MaxZ && og == 25){
                oggetti.set_luogo(og,0);
                interfaccia.scrivi("Lo zaino e' troppo pieno..."); 
                interfaccia.scrivi("Lo tengo in mano, e' troppo importante."); 
                interfaccia.scrivi("(Suggerimento: controlla nell'inventario)");
            } else{
                interfaccia.scrivi("Ora e' nello zaino.");
                z.inpila(oggetti.get_oggetto(og).get_codice()); //*
                n_oggettiZ++;
                peso_MaxZ -= oggetti.get_oggetto(og).get_peso();
            }
        } else
            avvisato = false;
    if(og == 23){
        stringa_risposta = "l'hai preso." //Modifica PMF(storia)
        storia_gioco.insStoria(stringa_comando , stringa_risposta); //Modifica PMF(storia)
        bacheca.CancellaMessaggioGioco
        ("*stai assorbendo troppe radiazioni! non hai la giusta protezione!");
    }else{
        avvisato = false;
    }

    return avvisato;
    //Fine Modifiche
}

```

```

void Astro::azione_142(){           //Azione indossa/metti
    if (oggetti.get Oggetto(og).get_luogo() == 0)
        // controllo che l'oggetto si trovi nell'inventario
        interfaccia.scrivi("- Gia' fatto.");
    else if (oggetti.get Oggetto(og).get_luogo() < 0)
        // controllo che l'oggetto non sia indossabile
        interfaccia.scrivi("- Non e' possibile.");
    else if (!indossa_specifiche()) {
        if(oggetti.get Oggetto(144).get_luogo() == 0){
            if (oggetti.get Oggetto(4).get_luogo() == 20 ||
                oggetti.get Oggetto(11).get_luogo() == 20 ||
                oggetti.get Oggetto(22).get_luogo() == 20){
                if(og == 4 || og == 11 || og == 23){
                    int c;
                    c=z.leggipila();
                    if(og == oggetti.get_zaino2(c)){
                        // controllo che l'oggetto sia proprio il primo oggetto nello zaino
                        oggetti.set_luogo(og,0);
                        z.fuoripila();
                        n_oggettiZ--;
                        peso_MaxZ += oggetti.get Oggetto(og).get_peso();
                    }
                } else if(oggetti.get Oggetto(og).get_luogo() == luogo_attuale){
                    // controllo che l'oggetto si trovi nel luogo attuale
                    oggetti.set_luogo(og,0);
                }
                else{
                    interfaccia.scrivi("Devi prima lasciare: ");
                    oggetti.get_zaino(c);
                }
            } else{
                oggetti.set_luogo(og,0);
                interfaccia.scrivi("Fatto.");
            }
        } else{
            oggetti.set_luogo(og,0);
        }
    } else if(oggetti.get Oggetto(143).get_luogo() == 0){
        if (oggetti.get Oggetto(4).get_luogo() == 20 ||
            oggetti.get Oggetto(11).get_luogo() == 20 ||
            oggetti.get Oggetto(23).get_luogo() == 20){
            if(og == 4 || og == 11 || og == 23){
                int c;
                c=og;
                if(ins.Appartiene(oggetti.get_valigia(c))){
                    // controllo che l'oggetto sia presente nella valigia
                    oggetti.set_luogo(og,0);
                    ins.Cancella(oggetti.get_valigia(c));
                    n_oggettiZV--;
                    peso_MaxZV += oggetti.get Oggetto(og).get_peso();
                }
            } else if(!ins.Appartiene(oggetti.get_valigia(c)))
                //controllo che l'oggetto non sia presente nella valigia
                interfaccia.scrivi("- Non ce l'hai in valigia.");
            else if(oggetti.get Oggetto(og).get_luogo() == luogo_attuale){
                //controllo che l'oggetto si trovi nel luogo attuale
                oggetti.set_luogo(og,0);
            }
        } else{
    }
}

```

```

        oggetti.set_luogo(og,0);
        interfaccia.scrivi("Fatto.");
    }
}
else{
    oggetti.set_luogo(og,0);
}
}
else{
    if(og != 0 && oggetti.get_oggetto(og).get_luogo() != 20){
        // controllo che l'oggetto non si trovi in un altro luogo e che l'oggetto
        non si trovi nello zaino o nella valigia
        oggetti.set_luogo(og,0);
    }
    else
        interfaccia.scrivi("- Sta nello zaino o nella valigia, e non li hai.");
}
if (!indosso_specifiche()){ // faccio alcuni controlli in indosso_specifiche
    if(oggetti.get_oggetto(og).get_luogo() == 0)
        //controllo che l'oggetto si trovi nell'inventario
        interfaccia.scrivi("Fatto.");
}
}
}
}

```

```

void Astro::azione_143(){           //Azione inventario valigia
    if(oggetti.get_oggetto(143).get_luogo() == 0){
        if (!ins.InsiemeVuoto()){
            interfaccia.scrivi("Inventario Valigia");
            interfaccia.scrivi("\nVedo: ");
            if(ins.Appartiene(50)) // controllo che il casco sia presente nella valigia
                oggetti.get_zaino(50);
            if(ins.Appartiene(51)) // controllo che la tua sia presente nella valigia
                oggetti.get_zaino(51);
            if(ins.Appartiene(52)) // controllo che il camice sia presente nella valigia
                oggetti.get_zaino(52);
            if(ins.Appartiene(55)) // controllo che il manuale sia presente nella valigia
                oggetti.get_zaino(55);
            cout << "\nTotale Oggetti nella Valigia: " << n_oggettiZV << endl;
            cout << "Spazio disponibile: " << peso_MaxZV << " su 8 kg." << endl;
        }
        else
            interfaccia.scrivi("E' vuota.");
    }
    else
        interfaccia.scrivi("Non ce l'hai.");
}

```

```

void Astro::azione_144(){           //Azione inventario zaino
    if(oggetti.get Oggetto(144).get_luogo() == 0){
        // controllo che l'oggetto zaino si trovi nell'inventario
        if (!z.pilavuota()) { // controllo che l'oggetto zaino non sia vuoto /**
            interfaccia.scrivi("Inventario Zaino");
            interfaccia.scrivi("\nVedo in cima: ");
            int c;
            c=z.leggipila(); /**
            oggetti.get_zaino(c);
            cout << "\nTotale Oggetti nello Zaino: " << n Oggettiz << endl;
            cout << "Spazio disponibile: " << peso_MaxZ << " su 5 kg." << endl;
        }
        else
            interfaccia.scrivi("E' vuoto.");
    }
    else
        interfaccia.scrivi("Non lo hai.");
}

void Astro::azione_145(){           //Azione aiuto/help
    interfaccia.scrivi("\nCOMANDI DI GIOCO:");
    interfaccia.scrivi("\nDirezioni: ");
    interfaccia.scrivi("- n/nord: per muoverti in avanti;");
    interfaccia.scrivi("- s/sud: per muoverti indietro;");
    interfaccia.scrivi("- e/est: per muoverti a destra;");
    interfaccia.scrivi("- w/o/ovest: per muoverti a sinistra;");
    interfaccia.scrivi("- a/alto/sali: per salire ad un piano superiore;");
    interfaccia.scrivi("- b/basso/scendi: per scendere ad un piano inferiore;");
    interfaccia.scrivi("\nAzioni: ");
    interfaccia.scrivi("- prendi/raccogli: per trasportare un oggetto in mano
                        o con se(zaino/valigia);");
    interfaccia.scrivi("- indossa/metti: per indossare un oggetto(es. casco);");
    interfaccia.scrivi("- guarda: per guardare ed ottenere informazioni su un oggetto
                        (es.tuta);");
    interfaccia.scrivi("- lascia/togli/leva: per lasciare o togliersi gli oggetti
                        trasportati;"); 
    interfaccia.scrivi("- apri: per aprire un oggetto fisso(es. armadietto);");
    interfaccia.scrivi("- leggi: per leggere una scritta(es.cartello);");
    interfaccia.scrivi("- spingi/tira: per spingere o tirare un oggetto
                        fisso(es.leva);");
    interfaccia.scrivi("- premi/schiaccia: per premere un oggetto
fisso(es.pulsante);");
    interfaccia.scrivi("- inventario/cosa: per accedere all'inventario degli oggetti
                        trasportati;"); 
    interfaccia.scrivi("- zaino: per accedere agli oggetti trasportati nello zaino;");
    interfaccia.scrivi("- valigia: per accedere agli oggetti trasportati nella
                        valigia;"); 
    interfaccia.scrivi("- save/load: per salvare o caricare la partita;");
    interfaccia.scrivi("- mappa/navigatore: per avviare il navigatore SIMUNAV;\n"); }

```

```
bool Astro::esegui_specifiche(int a, Mappa &M) -> sono stati aggiunti i seguenti case:
```

```
case 140:  
    azione_140();  
    break;  
  
case 141:  
    azione_141();  
    break;  
  
case 142:  
    azione_142();  
    break;  
  
case 143:  
    azione_143();  
    break;  
  
case 144:  
    azione_144();  
    break;  
  
case 145:  
    azione_145();  
    break;
```

5.17.3 Gioco.h

È stata aggiunta l'include per permettere l'implementazione della valigia:

```
#include "Valigia.h"
```

Sono state aggiunte le seguenti variabili:

```
Pila<int> p;  
Pila<int> z;  
int n_oggettiZ;      //Numero di oggetti nello zaino  
int peso_MaxZ;       //Peso massimo trasportabile nello zaino  
  
Insieme<int> ins;  
int n_oggettiZV;     //Numero di oggetti nella valigia  
int peso_MaxZV;      //Peso massimo trasportabile nella valigia  
  
//el1=primo elemento zaino; el2=secondo elemento zaino;  
//el3 [...] el6 elementi valigia  
int el1;  
int el2;  
int el3;  
int el4;  
int el5;  
int el6;  
bool salva;          //Booleano usato nella save e nella load  
string risposta1;   //risposta al frammento  
bool controllo1;  
bool controllo2;  
Lista<int> L
```

5.17.4 Gioco.cpp

Sono state modificate le seguenti funzioni:

```
void Gioco::prendi()
{
    //modifica Callone Gianmarco - Riadattamento metodo prendi per tasca, zaino e portafoglio.
    bool fatto = false;
    bool cpz= true ;
    riferimento_tasca = tasca.primolista();
    riferimento_zaino = zaino_frigo.primolista();
    bool trovato = false;
    int check = 0;

    if(og == 11 || og == 4 || og == 23 || og == 25)
        //controllo che gli oggetti che possono andare in valigia o nello zaino siano
        //o il casco, o la tuta, o il camice o il manuale
    {
        if (oggetti.get_oggetto(og).get_luogo() == 0)
            //controllo che l'oggetto non sia stato già preso
        {
            interfaccia.scrivi("- Gia' fatto.");
            stringa_risposta = "non e' stato possibile perche' avevi gia' quell'oggetto.";
            //Modifica PMF(storia)
            storia_gioco.insStoria(stringa_comando, stringa_risposta);
            //Modifica PMF(storia)
        }
        else if(oggetti.get_oggetto(og).get_luogo() == 20)
            //controllo che l'oggetto non sia già nella valigia o nello zaino
        {
            interfaccia.scrivi("- Sta gia' nello zaino o nella valigia.");
        }
        else if (oggetti.get_oggetto(og).get_luogo() < 0)
            //controllo che l'oggetto si possa prendere del tutto
        {
            stringa_risposta = "non e' stato possibile.";
            //Modifica PMF(storia)
            storia_gioco.insStoria(stringa_comando, stringa_risposta);
            //Modifica PMF(storia)
            interfaccia.scrivi("- Non e' possibile.");
        }
        else if (!prendi_specifiche())
        {
            controllo2 = false;
            controllo1 = false;
        }
    }
}
```

```

if(oggetti.get Oggetto(143).get Luogo() == 0)
// controllo se la valigia si trova nell'inventario
{
    if (og == 21 || og == 24 || og==49)
    {
        oggetti.set_Luogo(og,0);
    }
    else
    {
        do
        {
            interfaccia.scrivi(
                "Se vuoi portare l'oggetto con te, premi(y); se vuoi metterlo in valigia, premi(v)");

            cin >> rispostal;
            if(rispostal == "y" || rispostal == "Y")
            {
                oggetti.set_Luogo(og,0);
                interfaccia.scrivi("Ora e' nella valigia.");
                controllo1 = true;
                controllo2 = true;
            }
            else if(rispostal == "v" || rispostal == "V")
            {
                oggetti.set_Luogo(og,20);
                controllo2 = true;
            }
            else
            {
                controllo2 = false;
            }
        }
        while(controllo2 == false);
    }
}
else if(oggetti.get Oggetto(144).get Luogo() == 0)
{
    if (og == 21 || og == 24 || og==49)
    {
        oggetti.set_Luogo(og,0);
    }
    else
    {

```

```

do
{
    interfaccia.scrivi(
    "Se vuoi portare l'oggetto con te, premi(y); se vuoi metterlo nello zaino, premi(z)");
    cin >> rispostal;
    if(rispostal == "y" || rispostal == "Y")
    {
        oggetti.set_luogo(og,0);
        interfaccia.scrivi("Ora e' nell'inventario.");
        controllo1 = true;
        controllo2 = true;
    }
    else if(rispostal == "z" || rispostal == "Z")
    {

        oggetti.set_luogo(og,20);
        controllo2 = true;
    }
    else
    {
        controllo2 = false;
    }
}
while(controllo2 == false);
}

else
{
    oggetti.set_luogo(og,0);
}

if(!controllo1 && !preso_specifiche())
{
    interfaccia.scrivi("Fatto.");
    stringa_risposta = "l'hai preso.";
    //Modifica PMF(storia)
    storia_gioco.insStoria(stringa_comando, stringa_risposta);
    //Modifica PMF(storia)
}

bacheca.CancellaMessaggioGioco(
    "*non uscire fuori dall'astronave se non hai l'equipaggiamento da astronauta");
}
}

```

```

else if(!prendi_specifiche())
{
    if (oggetti.get_oggetto(og).get_luogo() == 0)
    {
        interfaccia.scrivi("- Già fatto.");
        stringa_risposta = "non è stato possibile perché avevi già quell'oggetto.";
        //Modifica PMF(storia)
        storia_gioco.insStoria(stringa_comando, stringa_risposta);
    }
    else if (oggetti.get_oggetto(og).get_luogo() < 0)
    {
        stringa_risposta = "non è stato possibile.";
        //Modifica PMF(storia)
        storia_gioco.insStoria(stringa_comando, stringa_risposta);
        //Modifica PMF(storia)
        interfaccia.scrivi("- Non è possibile.");
    }
    else if ((og != 27) && (og != 36) && (og != 37) && (og != 38) && (og != 39) &&
              (og != 40) && (og != 41) && (og != 42) && (og != 43) && (og != 44))
    {
        if(!preso_specifiche())
        {
            oggetti.set_luogo(og,0);
            if (og == 26)
            {
                interfaccia.scrivi("Fatto : l'hai addosso ! ");
                stringa_risposta = "l'hai preso.";
                //Modifica PMF(storia)
                storia_gioco.insStoria(stringa_comando, stringa_risposta);
                //Modifica PMF(storia)
            }
            else
            {
                interfaccia.scrivi("Fatto.");
                stringa_risposta = "l'hai preso.";
                //Modifica PMF(storia)
                storia_gioco.insStoria(stringa_comando, stringa_risposta);
                check++;
            }
        }
    }
    //Gestione portafogli
}
else if ( og == 27 || (og >=36 && og <= 44))
{

```

```

bool port = false;
port = portafoglio.hai_Portafoglio(oggetti);
if (port && (og>=36 && og <=44) )
{
    if(!fatto)
        portafoglio.Prendi_Banconota(oggetti,og,interfaccia);
    oggetti.set_luogo(og,0);
}
else if(!port && (og>=36 && og <=44))
    interfaccia.scrivi(
        " Non e' Possibile prendere denaro se non hai gia' preso il portafoglio ! ");
else if (!port && (og==27))
    interfaccia.scrivi(
        "Non e' Possibile prendere la carta di credito se non hai gia' preso il portafoglio ! ");
else if (port &&(og==27))
{
    oggetti.set_luogo(og,0);
    interfaccia.scrivi("Presa! L'hai nel portafoglio");
}
}
else if (og == 49 || og == 118 || og == 73)
//non puoi prendere: motorino
{
    interfaccia.scrivi ("Quest'oggetto non e' tuo. Per averlo dovresti rubarlo!");
    // chiave_auto
    interfaccia.a_capo();
    // buono pasto giallo
    oggetti.set_rubato(og, false);
}
else if(og == 74)      // Modifica documento d'identità //
{
    oggetti.set_luogo(og, 0);
    tasca.inslista(og, riferimento_tasca);
}
else if (og >=77 && og <= 79)
{
    oggetti.set_luogo(og, 0);
    tasca.inslista(og,riferimento_tasca);
    interfaccia.scrivi("Inserito nella tasca!");
}
else if (og >= 67 && og <=76)
{
    if (tasca.listavuota())
    {
        interfaccia.scrivi("Non hai buoni pasto per acquistarlo!");
        cpz=false;
    }
}

```

```

    else
    {
        interfaccia.scrivi("Dovresti comprare le pietanze, non prenderle... ");
        cpz=false;
    }
}

//INIZIO modifica PALAGIANO MARCELLO
else if(oggetti.get Oggetto(117).get_luogo() == 0)
{
    interfaccia.scrivi(" *** Per prendere oggetti devi lasciare l'autobus! *** ");
    cpz=false;
}
else if(oggetti.get Oggetto(119).get_luogo() == 0)
{
    interfaccia.scrivi(" *** Per prendere oggetti devi lasciare l'automobile! *** ");
    cpz=false;
}
//FINE modifica PALAGIANO MARCELLO
else
    oggetti.set_luogo(og,0);
//Fine Modifica Gallone
// MANCA NELLE VERSIONI INTERMEDIATE
if (!preso_specifiche() && (cpz) && check == 0)
{
    oggetti.set_rubato(og, false); // modifica CHIARAPPA ROSA
    interfaccia.scrivi("Fatto.");
    interfaccia.a_capo();
}
}

while (!tasca.finelista(riferimento_tasca))
{
    if (oggetti.get Oggetto(og).get_codice() ==
        oggetti.get Oggetto(tasca.leggilista(riferimento_tasca)).get_codice())
        trovato = true;
    riferimento_tasca = tasca.succlista(riferimento_tasca);
}

while (!zaino_frigo.finelista(riferimento_zaino))
{
    riferimento_zaino = zaino_frigo.succlista(riferimento_zaino);
    oggetti.set_rubato(og, false);
}
}
}

```

```

void Gioco::lascia()
{
    riferimento_tasca = tasca.primolista();
    riferimento_zaino = zaino_frigo.primolista();
    bool trovato = false;
    if (og != 44 && (luogo_attuale != 21 && luogo_attuale != 22))
    {
        if (og == 0 || oggetti.get_oggetto(og).get_luogo() != 0)
            interfaccia.scrivi("- Non ce l'hai.");
        else if (!lascia_specifiche())
        {
            if (og >= 77 && og <= 79)
                while (!tasca.finelist(riferimento_tasca) && trovato == false)
                {
                    if(oggetti.get_oggetto(tasca.leggilista(riferimento_tasca)).get_codice() ==
                       oggetti.get_oggetto(og).get_codice())
                    {
                        tasca.canclista(riferimento_tasca);
                        trovato = true;
                    }
                    else
                        riferimento_tasca = tasca.succlista(riferimento_tasca);
                }
            if (og >= 67 && og <= 76)
                while (!zaino_frigo.finelist(riferimento_zaino) && trovato == false)
                {
                    if(oggetti.get_oggetto(zaino_frigo.leggilista(riferimento_zaino)).get_codice() ==
                       oggetti.get_oggetto(og).get_codice())
                    {
                        zaino_frigo.canclista(riferimento_zaino);
                        trovato = true;
                    }
                    else
                        riferimento_zaino = zaino_frigo.succlista(riferimento_zaino);
                }
        }

        if (oggetti.get_oggetto(og).get_rubato()==true) //modifica ROSA CHIARAPPA (setto a false
                                                       // gli oggetti rubati lasciati)
        {
            oggetti.set_rubato(og, false);
        }

        oggetti.set_luogo(og, luogo_attuale);
        interfaccia.scrivi("Fatto.");
    }
}
else if (luogo_attuale != 12) //Modifica PMF(ufficio)
{

```

```

if(og == 0 && oggetti.get_oggetto(og).get_luogo() != 0 && oggetti.get_oggetto(og).get_luogo() != 20)
// controllo che l'oggetto si trovi in un altro luogo e che l'oggetto non si trovi né nell'inventario,
// né nella valigia, né nello zaino
{
    interfaccia.scrivi("- Non ce l'hai e qui non c'e'.");
    stringa_risposta = "non e' stato possibile perche' non avevi quell'oggetto.";//Modifica PMF(storia)
    storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
}

else if (!lascia_specifiche())
{
    // faccio alcuni controlli in lascia_specifiche
    if(oggetti.get_oggetto(144).get_luogo() == 0)
    {
        // controllo che lo zaino si trovi nell'inventario
        if(og==49) //se lascia il motorino
            salito=false;

        if (oggetti.get_oggetto(4).get_luogo() == 20 || oggetti.get_oggetto(11).get_luogo() == 20 ||
            oggetti.get_oggetto(23).get_luogo() == 20 || oggetti.get_oggetto(25).get_luogo() == 20)
        {
            // controllo che uno tra:tuta, casco, camice e manuale si trovi nello zaino
            if(og == 4 || og == 11 || og == 23 || og == 25)
            {
                // controllo che l'oggetto sia proprio uno tra: tuta, casco, camice e manuale
                if(oggetti.get_oggetto(og).get_luogo()== 0)
                {
                    // controllo che l'oggetto si trovi nell'inventario
                    oggetti.set_luogo(og,luogo_attuale);
                    if(og==49) //se lascia il motorino
                        salito=false;
                    interfaccia.scrivi("Fatto.");
                    stringa_risposta = "hai lasciato l'oggetto.";//Modifica PMF(storia)
                    storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
                }
                else
                {
                    // controllo che l'oggetto si trovi nello zaino
                    int c,d;
                    c=z.leggipila(); /*
                    d=og;
                    if(og == oggetti.get_zaino2(c))
                    // controllo che l'oggetto sia il primo visibile nello zaino
                    {
                        oggetti.set_luogo(og,luogo_attuale);
                        z.fuoripila(); /*
                        n_oggettiZ--;
                        peso_MaxZ += oggetti.get_oggetto(og).get_peso();
                        interfaccia.scrivi("Fatto.");
                    }
                    else if(ins.appartiene(oggetti.get_valigia(d)))
                    // controllo che l'oggetto si trovi nella valigia
                    {
                        ...
                    }
                }
            }
        }
    }
}

```

```

    {
        oggetti.set_luogo(og, luogo_attuale);
        interfaccia.scrivi("Fatto.");
        stringa_risposta = "hai lasciato l'oggetto." //Modifica PMF(storia)
        storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
    }
}
else
{
    if(oggetti.get_oggetto(og).get_luogo() != 0)
        // controllo che l'oggetto non si trovi nell'inventario
    {
        interfaccia.scrivi("- Non ce l'hai.");
        stringa_risposta = "non e' stato possibile perche' non avevi quell'oggetto." //Modifica PMF(storia)
        storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
    }
    else
    {
        oggetti.set_luogo(og, luogo_attuale);
        interfaccia.scrivi("Fatto.");
        stringa_risposta = "hai lasciato l'oggetto." //Modifica PMF(storia)
        storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
    }
}
else
{
    if(oggetti.get_oggetto(og).get_luogo() == 0)
        // controllo che l'oggetto si trovi nell'inventario
    {
        oggetti.set_luogo(og, luogo_attuale);
        if(og==39) //se lascio il motorino
            salito=false;
        interfaccia.scrivi("Fatto.");
        stringa_risposta = "hai lasciato l'oggetto." //Modifica PMF(storia)
        storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
    }
    /*else
        interfaccia.scrivi("- Non ce l'hai.");*/
    }
}
else if ((luogo_attuale == 7 || luogo_attuale >= 9) && (oggetti.get_oggetto(4).get_luogo() != 0 || oggetti.get_oggetto(11).get_luogo() != 0))
{
    // controllo che il luogo attuale sia o 7 o qualsiasi luogo pari o superiore a 9 e che la tutta o il casco non si trovino nell'inventario
    interfaccia.scrivi("\nAaaaagh!!!!");
    morto();
}
if (og == 67)
{
    interfaccia.scrivi("Non puoi compiere questa azione!");
}
} //Fine modifica
}

```

```
//Modifica PMF(ufficio)
else
{
    interfaccia.scrivi("Non puoi lasciarlo nell' ufficio.");
}
//Modifica PMF: fin qui.
```

```

void Gioco::save()
{
    int i;

    interfaccia.scrivi("Salvataggio Raktita...");
    ofstream file(fStringa.c_str(), ios::out);
    for (i = 1; i <= oggetti.get_n_oggettiZ(); i++)
    {
        file << oggetti.get_oggetto(i).get_luogo() << '\n';
    }
    file << luogo_attuale << '\n';
    file << tempo << '\n';
    file << passo_soluzione << '\n';

    //Modifiche_ML
    file << n_oggettiZ << '\n';
    file << peso_MaxZ << '\n';

    if(!z.pilavuota())           // controllo che lo zaino non sia vuoto
    {
        ell = z.leggipila();
        z.fuoripila();
        if(!z.pilavuota())       // controllo che lo zaino non sia vuoto
        {
            el2 = z.leggipila();
            z.fuoripila();
            z.inpila(el2);
        }
        z.inpila(ell);
    }
    file << ell << '\n';
    file << el2 << '\n';

    file << n_oggettiZV << '\n';
    file << peso_MaxZV << '\n';
    if(!ins.insiemevuoto())      //controllo che la valigia non sia vuota
    {
        if(ins.appartiene(50))   // controllo che il casco sia presente nella valigia
        {
            el3 = 50;
        }
        file << el3 << '\n';
        if(ins.appartiene(51))   // controllo che la tuta sia presente nella valigia
        {
            el4 = 51;
        }
        file << el4 << '\n';
        if(ins.appartiene(52))   // controllo che il camice sia presente nella valigia
        {
            el5 = 52;
        }
        file << el5 << '\n';
        if(ins.appartiene(55))   // controllo che il manuale sia presente nella valigia
        {
            el6 = 55;
        }
    }
}

```

```
    file << el6 << '\n';
}

//Fine modifiche
save_specifiche(file);
salvaStatiDialoghi(); //modifica D'Andria Dossola, metodo per salvare gli stati su file
bacheca.SalvaBacheca(file);
salvaJukeBoxLuci(); //modifica VACCA - Salva lo stato del Jukebox e delle luci
file.close();
//INIZIO MODIFICHE CICALA GIACOMO
for(int k=1; k<=31; k++)
    if(slotmachine.appartiene(k))
        numEuro++;

file << numEuro << '\n';

for(int j=1; j<=31; j++)
    if(slotmachine.appartiene(j))
        file << j << '\n';
//FINE MODIFICHE CICALA GIACOMO
}
```

```

void Gioco::load()
{
    int i;
    int valore;
    int slot; //CICALA GIACOMO
    svuotaInsieme(slotmachine); //CICALA GIACOMO

    ifstream file(fStringa.c_str(), ios::in);
    //modifica effettuata da D'Andria Donda sul controllo del caricamento da file
    ifstream astrostatidialoghi("Astrostatidialoghi.txt");
    bool astrostatidialoghiaperto=astrostatidialoghi.good();
    bool fileaperto=file.good();
    if(fileaperto || astrostatidialoghiaperto)
    {

        interfaccia.scrivi("Ripristino partita...");
        if(fileaperto)
        {
            for (i = 1; i <= oggetti.get_n_oggetti(); i++)
            {
                file >> valore;
                oggetti.set_luogo(i, valore);
            }
            file >> luogo_attuale;
            file >> tempo;
            file >> passo_soluzione;

            while(! (z.pilavuota()))
                z.fuoripila(); //controllo che lo zaino non sia vuoto
            file >> n_oggettiZ;
            file >> peso_MaxZ;
            file >> ell;
            file >> el2;
            if(el2 !=0)
                z.inpila(el2);
            if(ell !=0)
                z.inpila(ell);

            file >> n_oggettiZV;
            file >> peso_MaxZV;
            file >> el3;
            file >> el4;
            file >> el5;
            file >> el6;
            if((! (ins.appartiene(el3))) && el3 != 0)
                ins.inserisci(el3);
            if((! (ins.appartiene(el4))) && el4 != 0)
                ins.inserisci(el4);
            if((! (ins.appartiene(el5))) && el5 != 0)
                ins.inserisci(el5);
            if((! (ins.appartiene(el6))) && el6 != 0)
                ins.inserisci(el6);
        }
        //Fine ML
    }
}

```

```

load_specifiche(file);
bacheca.CaricaBacheca(file);
file.close();
}
if(astrostatidialoghiaperto)
{
    caricaStatiDialoghi(); //modifiche D'Andria Dresden
}
ifstream jukeLuci("JukeLuci.txt");
if (jukeLuci.good())
{
    caricaJukeBoxLuci();
}

}
else
{
    interfaccia.scrivi("Non ci sono partite salvate!");
}
//INIZIO MODIFICHE CICALA GIACOMO
file >> numEuro;

for(int j=1; j<=numEuro; j++)
{
    file >> slot;
    slotmachine.inserisci(slot);
}
//fine modifiche CICALA GIACOMO
}

```

5.17.5 Mappa.h

È stata aggiunta l'include per permettere l'implementazione dello zaino:

```
#include "Zaino.h"
```

5.17.6 Interfaccia.cpp

È stata modificata la seguente funzione:

```
void Interfaccia::elenca_oggetti(Oggetti oggetti, string premessa)
{
    //Modifica Mirco Sternativo
    int n_oggettiZ = oggetti.get_n_oggettiZ();

    if (n_oggettiZ > 0)
    {
        cout << premessa;
        for (int i = 1; i <= n_oggettiZ; i++)
        {
            cout << "\n- " << oggetti.get Oggetto(i).get nome();

            if (i == n_oggettiZ)
            {
                cout << ".";
            }
            else
            {
                cout << ",";
            }
        }
        cout << endl;
    }
    //fine modifiche
}
```

5.17.7 Oggetti.h

Sono stati aggiunti i seguenti metodi:

```
int get_n_oggettiZ();
void get_zaino(int);
int get_zaino2(int);
int get_valigia(int);
```

5.17.8 Oggetti.cpp

Sono stati aggiunti i seguenti metodi:

```
int Oggetti::get_n_oggettiZ()
{
    return fo;
}

void Oggetti::get_zaino(int c)
{
    bool trovato = false;
    int i = 1;
    while (i <= fo && !trovato) {
        // scandisco tutti gli oggetti presenti nel vocabolario finché o
        // finiscono gli oggetti o trovato viene impostato a true
        if (oggetti[i].get_codice() == c){
            // controllo che il codice dell'oggetto con indice "i" sia uguale al
            // codice "c"
            cout << " - " << oggetti[i].get_nome() << endl;
            trovato = true;
        }
        i++;
    }
}
```

```
int Oggetti::get_zaino2(int c)
```

```

{
    bool trovato = false;
    int i = 1;
    int i2 = 0;
    while (i <= fo && !trovato) {
        // scandisco tutti gli oggetti presenti nel vocabolario finché o
        finiscono gli oggetti o trovato viene impostato a true
        if (oggetti[i].get_codice() == c){
            // controllo che il codice dell'oggetto con indice "i" sia uguale al
            codice "c"
            i2=i;
            trovato = true;
        }
        i++;
    }
    return(i2);
}

int Oggetti::get_valigia(int c)
{
    bool trovato = false;
    int i = 1;
    int i2 = 0;
    while (i <= fo && !trovato) {
        // scandisco tutti gli oggetti presenti nel vocabolario finché o
        finiscono gli oggetti o trovato viene impostato a true
        if (i == c){//controllo che l'indice "i" sia uguale al codice "c"
            i2 = oggetti[i].get_codice();
            trovato = true;
        }
        i++;
    }
    return(i2);
}

```

Ed è stata modificata la seguente funzione:

```
int Oggetti::luogo_oggetto(int c2, int lu) {
    bool trovato = false;
    int i = 1;
    int og = 0;

    while (i <= fo && !trovato) {
        if (c2 == 30){
            if(oggetti[i].get_luogo() == lu){
                og = i;
            }
        }else if (oggetti[i].get_codice() == c2)
            if (abs(oggetti[i].get_luogo()) == lu || oggetti[i].get_luogo() ==
0 || oggetti[i].get_luogo() == 20) {
                og = i;
                trovato = true;
            }
        i++;
    }
    return og;
}
```

5.17.9 Oggetto.h

È stato creato un nuovo costruttore che integra l'attributo peso

```
Oggetto(string n, int c, int l, int w);
//Modifica Mirco Sternativo -- Peso oggetto
```

È stato modificato il costruttore avente l'attributo prezzo, aggiungendo il peso

```
Oggetto(string n, int c, int l, int w, float p );
//MODIFICA D-R(D'Orsi):Negozio + Banca
//Modifica Mirco Sternativo -- Peso oggetto
```

È stato modificato il metodo get_peso:

```
int get_peso(); //Modifica Mirco Sternativo
```

5.17.10 Oggetto.cpp

È stato aggiunto il seguente metodo

```
Oggetto::Oggetto(string n, int c, int l, int w)      //w: weight
{
    nome = n;
    codice = c;
    luogo = l;
    peso = w;
}
```

Sono stati modificati i seguenti metodi

```
Oggetto::Oggetto(string n, int c, int l, int w, float p)
{
    nome = n;
    codice = c;
    luogo = l;
    peso = w;
    prezzo = p;
}

int Oggetto::get_peso()
{
    return peso;
}
```

5.17.11 Zaino.h

È stato semplicemente aggiunto il file senza alcuna modifica particolare.

5.17.12 Valigia.h

È stato semplicemente aggiunto il file senza alcuna modifica particolare.

6 STRUTTURE INTERCAMBIABILI

Sono state inserite le strutture intercambiabili:

6.1 LISTA

6.1.1 Lista con Puntatore

```
#ifndef _LISTAMONO_H

#define _LISTAMONO_H

#include <iostream>

#include <cstdlib>

#include "Cella.h"

using namespace std;

template<class tipoelem> class Lista{

public:

    typedef Cella<tipoelem>* posizione;

    Lista();
    Lista(const Lista<tipoelem>&);

    ~Lista();

    void crealista();

    bool listavuota() const;

    bool finelista(posizione) const;

    posizione primolista() const;

    posizione succlistा( posizione) const;

    posizione preclista( posizione) const;

    tipoelem leggilista( posizione) const;

    void scrivilista(tipoelem, posizione);

    void inslista(tipoelem, posizione&);

    void canclista(posizione&);

    //Operatori ausiliari

    void svuota();

    void stampaLista(); //Modifica Zagaria -- aggiunta funzione stampalista

    Cella<tipoelem>* get_posizione()const; // MODIFICA D-R(D-R)
```

```

//sovraffabbrichi

void operator =(const Lista&);

bool operator ==(Lista&);

private:

    posizione lista;

    unsigned int lungh;

    unsigned int lunghezza(); //Operatore ausiliario che restituisce la lunghezza della lista

};

template <class tipoelem> Lista<tipoelem>::Lista()

{

    crealista();

}

template <class tipoelem>Lista<tipoelem>::Lista(const Lista& b)

{

    crealista();

    typename Lista<tipoelem>::posizione ind = primolista(),ind2 = b.primolista();

    while (!b.finlista(ind2))

    {

        inslista(b.leggilista(ind2),ind);

        ind = succlista(ind);

        ind2 = b.succlista(ind2);

    }

}

template <class tipoelem> Lista<tipoelem>::~Lista() //distruttore

{

    posizione ind=primolista();

    while (!finlista(ind))

    {

        canclista(ind);

    }

    delete lista;

}

template <class tipoelem> void Lista<tipoelem>::crealista()

```

```

{
    lista=new Cella<tipoelem>;
    lista->scrivisucc(nullptr);
    lungh=0;
}

template <class tipoelem> bool Lista<tipoelem>::listavuota() const
{
    return (lista->leggisucc()==nullptr);
}

template <class tipoelem> bool Lista<tipoelem>::finelista(posizione p) const
{
    return (p->leggisucc()==nullptr);
}

template <class tipoelem> Cella<tipoelem>* Lista<tipoelem>::primolista() const
{
    return (lista);
}

template <class tipoelem> Cella<tipoelem>* Lista<tipoelem>::succlista(posizione p) const
{
    if (!finelista(p))
        return (p->leggisucc());
    else return (p);
}

template <class tipoelem> Cella<tipoelem>* Lista<tipoelem>::preclista(posizione p) const
{
    if (p!=primolista() && !finelista(p))
    {
        posizione ind=primolista();
        while (ind->leggisucc()!=p)
        {
            ind=ind->leggisucc();
        }
    }
}

```

```

        return(ind);
    }
    else return (p);
}

template <class tipoelem> tipoelem Lista<tipoelem>::legglista(posizione p) const
{
    if (!finelista(p))
        return (p->leggicella());
}

template <class tipoelem> void Lista<tipoelem>::scrivilista(tipoelem e, posizione p)
{
    if (!finelista(p))
        p->scrivicella(e);
}

template <class tipoelem> void Lista<tipoelem>::inslista(tipoelem e, posizione& p)
{
    typename Lista<tipoelem>::posizione temp;
    temp = new Cella<tipoelem>;
    temp->scrivicella(e);
    temp->scrivisucc(p);
    temp->scriviprec(p->leggiprec());
    if(p == primolista())
        lista = temp;
    else
        p->leggiprec()->scrivisucc(temp);
    p->scriviprec(temp);
    p = temp;
}

template <class tipoelem> void Lista<tipoelem>::canclista(posizione &p)
{
    if (!finelista(p))
    {

```

```

posizione temp=p;
if (p==lista)
{
    lista=p->leggisucc();
    p=lista;
}
else
{
    p=p->leggisucc();
    (preclista(temp))->scrivisucc(p);
}
delete(temp);
lungh--;
}

}

//MODIFICA D-R(D-R) //
template <class tipoelem> Cella<tipoelem>* Lista<tipoelem>::get_posizione()const
{
    return lista;
}

// sovraccarico

template <class tipoelem> void Lista<tipoelem>::operator =(const Lista<tipoelem>& l) //sovraccarico
operatore di assegnamento
{
    if (this!=&l) //non posso assegnare per esempio A=A;
    {
        svuota(); //svuoto
        posizione ind=primolista();
        posizione ind2=l.primolista();
        while (!l.finelista(ind2))
        {
//            inslista(ind,l.leggilista(ind2)); //e inserisco in essa tutti gli elementi di l
        }
    }
}

```

```

        ind=succlista(ind);
        ind2=l.succlista(ind2);
    }
}

template <class tipoelem> bool Lista<tipoelem>::operator ==(Lista<tipoelem>& l) //sovraffaccio operatore
di uguaglianza

{
    bool uguali=lunghezza()==l.lunghezza(); //vedo le lunghezze delle liste
    if (uguali) //se le liste hanno la stessa lunghezza
    {
        posizione ind=primolista(); //---+
        // +----- mi posiziono all'inizio di entrambe le liste
        posizione ind2=l.primolista(); //+-
        while (!finelista(ind) && uguali) // mentre non ho finito le liste e tutti i rispettivi caratteri delle lista
sono uguali
        {
            uguali=(uguali && (legglista(ind)==l.legglista(ind2))); //uguali sarà dato dal valore precedente di
uguali AND il risultato del confronto dei simboli correnti
            ind=succlista(ind); //se almeno uno dei caratteri è diverso avrà valore false
            ind2=l.succlista(ind2);
        }
        return (uguali);
    }
}

template<class tipoelem> ostream& operator<<(ostream& os, const Lista<tipoelem>& l) //sovraffaccio
output
{
    Cella<tipoelem>* ind=l.primolista();
    while (!l.finelista(ind))
    {
        os<< l.legglista(ind);
        ind=l.succlista(ind);
        if (!l.finelista(ind)) os<< ",";
    }
}

```

```

    return(os);
}

// operatori ausiliari

template <class tipoelem> void Lista<tipoelem>::svuota() //svuota la lista
{
    posizione indice=primolista();

    while (!listavuota())
    {
        canclista(indice); //mentre la lista non è vuota cancella il primo elemento
    }
}

template <class tipoelem> unsigned int Lista<tipoelem>::lunghezza() //restituisce la lunghezza della lista
{
    return(lungh);
}

// inizio modifica Zagaria -- aggiunto corpo funzione stampalista

template <class tipoelem> void Lista<tipoelem>::stampaLista ()
{
    posizione p;
    p = primolista();
    while (!finelista(p))
    {
        cout<<leggilista(p);
        p = succlist(a(p));
    }
}

// fine modifiche zagaria
#endif

```

6.1.2 Lista con Vettore sequenziale

```
#ifndef _LISTA_H
```

```
#define _LISTA_H
```

```
template <class T>
```

```
class Lista{
public:
    typedef int posizione;
    typedef T tipoelem;
    Lista ();
    ~Lista ();
    void crealista ();
    bool listavuota () const;
    tipoelem leggilista (posizione)const;
    void scrivilista (tipoelem,posizione &);
    posizione primolista () const;
    bool finelista (posizione)const;
    posizione succlistा (posizione)const;
    posizione preclista (posizione)const;
    void inslista (tipoelem,posizione &);
    void canclista (posizione &);

private:
    static const int max = 1024;
    T * elementi;
    int lunghezza;
};

template <class T>Lista <T>::Lista()
{
    crealista();
}

template <class T>Lista <T>::~Lista()
{
    lunghezza = 0;
    delete elementi;
}

template <class T>void Lista <T>::crealista()
{
}
```

```

lunghezza = 0;
elementi = new T[max];
}

template <class T>bool Lista <T>::listavuota() const
{
    return (lunghezza == 0);
}

template <class T>typename Lista <T>::posizione Lista <T>::primolista() const
{
    return 0;
}

template <class T>typename Lista <T>::posizione Lista <T>::succlista(posizione p)const
{
    if (0 <= p && p < lunghezza)
        return (p + 1);
    else
        return p;
}

template <class T>typename Lista <T>::posizione Lista <T>::preclista(posizione p)const
{
    if (0 < p && p < lunghezza)
        return (p - 1);
    else
        return p;
}

template <class T>bool Lista <T>::finelista(posizione p) const
{
    if (0 <= p && p <= lunghezza)
        return (p == lunghezza);
    else
        return false;
}

```

```

template <class T>T Lista <T>::legglista(posizione p)const
{
    return elementi[p];
}

template <class T>void Lista <T>::scrivilista(tipoelem a, posizione & p)
{
    elementi[p] = a;
}

template <class T>void Lista <T>::inslista(tipoelem a, posizione & p)
{
    for(int i = lunghezza; i > p; i--)
        elementi[i] = elementi[i-1];
    elementi[p] = a;
    lunghezza++;
}

template <class T>void Lista <T>::canlista(posizione & p)
{
    for(int i=p; i < (lunghezza-1); i++)
        elementi[i] = elementi[i+1];
    lunghezza--;
}

#endif // _LISTA_H

```

6.1.3 Lista con puntatori doppi

```

1  /*
2   * Realizzazione: Lista con Puntatori Doppi
3   * Autore: A. Annese
4   * Nota Bene: Il costruttore di copia e' stato implementato in quanto
5   *             NECESSARIO per via di alcuni file del progetto in cui
6   *             vengono passate liste come parametri di funzioni e
7   *             restituite (dalle stesse) non per riferimento.
8   */
9  #ifndef LISTA_H_
10 #define LISTA_H_
11
12 #include "Cella_L_DP.h"

```

```

13
14  template <class tipoelem>
15  class Lista{
16  public:
17      typedef Cella_L_DP<tipoelem>* posizione;
18
19      Lista();
20      Lista(const Lista&);
21      ~Lista();
22
23      void crealista();
24
25      bool listavuota() const;
26      posizione primolista() const;
27      posizione succlista(posizione) const;
28      posizione preclista(posizione) const;
29      bool finelista(posizione) const;
30
31      tipoelem leggilista(posizione) const;
32
33      void inslista(tipoelem, posizione&);
34      void scrivilista(tipoelem, posizione);
35      void canclista(posizione&);
36
37  private:
38      posizione lista;
39  };
40
41  template <class tipoelem>
42  Lista<tipoelem>::Lista()
43  {
44      crealista();
45  }
46
47  template <class tipoelem>
48  Lista<tipoelem>::Lista(const Lista& b)
49  {
50      crealista();
51      typename Lista<tipoelem>::posizione ind = primolista(),ind2 = b.primolista();
52      while (!b.finelista(ind2))
53      {
54          inslista(b.leggilista(ind2),ind);
55          ind = succlista(ind);
56          ind2 = b.succlista(ind2);
57      }
58  }
59
60  template <class tipoelem>
61  Lista<tipoelem>::~Lista()
62  {
63      posizione p = primolista();
64      posizione temp;
65      while(p->getSucc() != nullptr)
66      {
67          temp = p;
68          p=p->getSucc();
69          delete temp;
70      }
71  }
72
73  template <class tipoelem>
74  void Lista<tipoelem>::crealista()
75  {
76      lista = new Cella_L_DP<tipoelem>;
77      lista->setSucc(nullptr);
78      lista->setPrec(nullptr);
79  }

```

```

80
81     template <class tipoelem>
82     bool Lista<tipoelem>::listavuota() const
83     {
84         return (lista->getSucc() == nullptr && lista->getPrec() == nullptr);
85     }
86
87     template <class tipoelem>
88     typename Lista<tipoelem>::posizione Lista<tipoelem>::primolista() const
89     {
90         return lista;
91     }
92
93     template <class tipoelem>
94     typename Lista<tipoelem>::posizione Lista<tipoelem>::succlista(posizione p) const
95     {
96         if(lista->getSucc() == nullptr)
97             return p;
98         else
99             return p->getSucc();
100    }
101
102    template <class tipoelem>
103    typename Lista<tipoelem>::posizione Lista<tipoelem>::preclista(posizione p) const
104    {
105        return p->getPrec();
106    }
107
108    template <class tipoelem>
109    bool Lista<tipoelem>::finelista(posizione p) const
110    {
111        return (p->getSucc() == nullptr);
112    }
113
114
115    template <class tipoelem>
116    tipoelem Lista<tipoelem>::legglista(posizione p) const
117    {
118        return p->getElem();
119    }
120
121    template <class tipoelem>
122    void Lista<tipoelem>::scrivilista(tipoelem e, posizione p)
123    {
124        p->setElem(e);
125    }
126
127    template <class tipoelem>
128    void Lista<tipoelem>::inslista(tipoelem e, posizione& p)
129    {
130        typename Lista<tipoelem>::posizione temp;
131
132        temp = new Cella_L_DP<tipoelem>;
133        temp->setElem(e);
134        temp->setSucc(p);
135        temp->setPrec(p->getPrec());
136
137        if(p == primolista())
138            lista = temp;
139        else
140            p->getPrec()->setSucc(temp);
141
142        p->setPrec(temp);
143
144        p = temp;
145    }
146

```

```

147  template <class tipoelem>
148  void Lista<tipoelem>::canclista(posizione& p)
149  {
150      posizione temp;
151      temp = p;
152      if(p == primolista())
153      {
154          if(p->getSucc() != nullptr)
155          {
156              lista = p->getSucc();
157              lista->setPrec(nullptr);
158          }
159      }
160      else
161      {
162          prelista(p)->setSucc(p->getSucc());
163          succlista(p)->setPrec(prelista(p));
164      }
165
166      p = p->getSucc();
167      delete temp;
168  }
169
170
171 #endif //LISTA_H_

```

6.1.3.1 Cella_L_DP.h

```

1  /*
2  Template: Cella per Lista con Doppi Puntatori
3  Autore: A. Annese
4  */
5 #ifndef CELLA_L_DP_H_
6 #define CELLA_L_DP_H_
7
8 template <class tipoelem>
9 class Cella_L_DP{
10 public:
11     Cella_L_DP();
12     ~Cella_L_DP();
13
14     tipoelem getElement() const;
15     void setElement(tipoelem);
16     Cella_L_DP<tipoelem>* getSucc() const;
17     void setSucc(Cella_L_DP<tipoelem>* );
18     Cella_L_DP<tipoelem>* getPrec() const;
19     void setPrec(Cella_L_DP<tipoelem>* );
20 private:
21     tipoelem elem;
22     Cella_L_DP<tipoelem>* succ;
23     Cella_L_DP<tipoelem>* prec;
24 };
25
26 template <class tipoelem>
27 Cella_L_DP<tipoelem>::Cella_L_DP()
28 {
29     succ = nullptr;
30     prec = nullptr;
31 }
32
33 template <class tipoelem>
34 Cella_L_DP<tipoelem>::~Cella_L_DP(){}
35
36 template <class tipoelem>
37 tipoelem Cella_L_DP<tipoelem>::getElement() const

```

```

38     {
39         return elem;
40     }
41
42     template <class tipoelem>
43     void Cella_L_DP<tipoelem>::setElem(tipoelem e)
44     {
45         elem = e;
46     }
47
48     template <class tipoelem>
49     Cella_L_DP<tipoelem>* Cella_L_DP<tipoelem>::getSucc() const
50     {
51         return succ;
52     }
53
54     template <class tipoelem>
55     void Cella_L_DP<tipoelem>::setSucc(Cella_L_DP<tipoelem>* e)
56     {
57         succ = e;
58     }
59
60     template <class tipoelem>
61     Cella_L_DP<tipoelem>* Cella_L_DP<tipoelem>::getPrec() const
62     {
63         return prec;
64     }
65
66     template <class tipoelem>
67     void Cella_L_DP<tipoelem>::setPrec(Cella_L_DP<tipoelem>* e)
68     {
69         prec = e;
70     }
71
72 #endif //CELLA_L_DP_H

```

6.2 LISTA ORDINATA

6.2.1 Lista Ordinata con vettore:

6.2.1.1 *ListaOrdinata.h*

```

1 //Realizzazione lista ordinata con vettore - MARCELLO PALAGIANO
2 #ifndef _LISTAORDINATA_H
3 #define _LISTAORDINATA_H
4 #include <assert.h>
5 template<class T>
6 class ListaOrdinata{
7 public:
8     typedef T tipoelem;
9     typedef int posizione;
10    //Costruttori
11    ListaOrdinata();
12    //Distruttori
13    ~ListaOrdinata();
14    //Operatori di specifica
15    void crealista();
16    bool listavuota() const;
17    tipoelem leggilista(posizione) const;
18    posizione primolista() const;
19    bool finalista(posizione) const;
20    posizione succlista(posizione) const;

```

```

21     posizione predlista(posizione) const;
22     void inslista(tipoelem);
23     void canclista(posizione);
24 private:
25     static const int MAXLUNG = 1024;
26     tipoelem vett[MAXLUNG];
27     unsigned int lunghezza;
28 };
29 template <class T>
30 ListaOrdinata<T>::ListaOrdinata(){
31     crealista();
32 }
33 template <class T>
34 ListaOrdinata<T>::~ListaOrdinata(){
35 }
36 template <class T>
37 void ListaOrdinata<T>::crealista(){
38     lunghezza=0;
39 }
40 template <class T>
41 bool ListaOrdinata<T>::listavuota() const{
42     return(lunghezza==0);
43 }
44 template<class T>
45 typename ListaOrdinata<T>::posizione ListaOrdinata<T>::primolista() const{
46     return(1);
47 }
48 template <class T>
49 bool ListaOrdinata<T>::finelista(posizione p) const{
50     assert((p>0)&&(p<=lunghezza+1));
51     return(p==lunghezza+1);
52 }
53 template <class T>
54 typename ListaOrdinata<T>::posizione ListaOrdinata<T>::succlista(posizione p) const{
55     assert((p>0 && p<lunghezza+1));
56     return(p+1);
57 }
58 template <class T>
59 typename ListaOrdinata<T>::posizione ListaOrdinata<T>::predlista(posizione p) const{
60     assert((p>1 && p<lunghezza+1));
61     return(p-1);
62 }
63 template <class T>
64 T ListaOrdinata<T>::legglista(posizione p) const{
65     assert((p>0 && p<lunghezza+1));
66     return(vett[p-1]);
67 }
68 template <class T>
69 void ListaOrdinata<T>::inslista(tipoelem e){
70     assert((lunghezza<MAXLUNG));
71     posizione p=primolista();
72     while(!finelista(p) && legglista(p)<e){
73         p=succlista(p);
74     }
75     for(int i=lunghezza; i>=p; i--){
76         vett[i]=vett[i-1];
77     }
78     vett[p-1]=e;
79     lunghezza++;
80 }
81 template <class T>
82 void ListaOrdinata<T>::canclista(posizione p){
83     assert(((p>0)&&(p<=lunghezza)));
84     for(int i=p-1; i<lunghezza; i++){
85         vett[i]=vett[i+1];
86         lunghezza--;
87 }

```

```
88     }
89 #endif // _LISTAORDINATA_H
```

6.2.2 Lista Ordinata con puntatori:

6.2.2.1 *ListaOrdinata.h*

```
1 //Realizzazione lista ordinata con puntatori - MARCELLO PALAGIANO
2 #ifndef _LISTAORDINATA_H
3 #define _LISTAORDINATA_H
4 #include <assert.h>
5 #include "cella.h"
6 template<class T>
7 class ListaOrdinata{
8 public:
9     typedef T tipoelem;
10    typedef Cella<T>* posizione;
11    //Costruttori
12    ListaOrdinata();
13    //Distruttori
14    ~ListaOrdinata();
15    //Operatori di specifica
16    void crealista();
17    bool listavuota() const;
18    bool finelista(posizione) const;
19    posizione primolista() const;
20    posizione succlistा( posizione) const;
21    posizione preclistा( posizione) const;
22    tipoelem leggilista( posizione) const;
23    void inslista(tipoelem);
24    void canclista(posizione);
25 private:
26     posizione lista;
27 };
28 template<class T>
29 ListaOrdinata<T>::ListaOrdinata(){
30     crealista();
31 }
32 template <class T>
33 ListaOrdinata<T>::~ListaOrdinata()
34 {
35     posizione p = primolista();
36     while(p->leggisucc() != NULL)
37     {
38         posizione temp = p;
39         p=p->leggisucc();
40         delete temp;
41     }
42 }
43 template<class T>
44 void ListaOrdinata<T>::crealista(){
45     lista=new Cella<T>;
46     lista->scrivisucc(NULL);
47 }
48 template<class T>
49 bool ListaOrdinata<T>::listavuota() const{
50     return(lista->leggisucc()==NULL);
51 }
52 template<class T>
53 bool ListaOrdinata<T>::finelista(posizione p) const{
54     return(p->leggisucc()==NULL);
55 }
56 template<class T>
57 typename ListaOrdinata<T>::posizione ListaOrdinata<T>::primolista() const{
58     return(lista);
59 }
```

```

60     template<class T>
61     typename ListaOrdinata<T>::posizione ListaOrdinata<T>::succlista(posizione p) const{
62         assert(!(p->leggisucc()==NULL));
63         return(p->leggisucc());
64     }
65     template<class T>
66     typename ListaOrdinata<T>::posizione ListaOrdinata<T>::preclista(posizione p) const{
67         typename ListaOrdinata<T>::posizione temp=primolista();
68         assert(!(p==temp));
69         while(succlista(temp)!=p){
70             temp=succlista(temp);
71         }
72         return(temp);
73     }
74     template<class T>
75     T ListaOrdinata<T>::legglista(posizione p) const{
76         return(p->leggicella());
77     }
78     template<class T>
79     void ListaOrdinata<T>::canclista(posizione p){
80         typename ListaOrdinata<T>::posizione temp=p;
81         assert(!(listavuota()));
82         if(p!=lista){
83             preclista(p)->scrivisucc(p->leggisucc());
84         }
85         else{
86             lista=p->leggisucc();
87         }
88         p=p->leggisucc();
89         delete temp;
90     }
91     template <class T>
92     void ListaOrdinata<T>::inslista(tipoelem e){
93         typename ListaOrdinata<T>::posizione temp;
94         typename ListaOrdinata<T>::posizione p=primolista();
95         while(!finelista(p) && legglista(p)<e){
96             p=succlista(p);
97         }
98         temp = new Cella<T>;
99         temp->scrivicella(e);
100        temp->scrivisucc(p);
101        if (p == primolista())
102            lista = temp;
103        else
104            preclista(p)->scrivisucc(temp);
105        p = temp;
106    }
107 #endif // _LISTAORDINATA_H

```

6.3 PILA

6.3.1 Pila con puntatori:

6.3.1.1 *Pila.h*

```

1. /**
2.  * @file Pila.h
3.  * Definizione della struttura dati "Pila".
4.  * @note Definizione di Pila per la realizzazione tramite puntatori.

```

```

5.   * @author Sconosciuto, modificato da Niccolo' Petti.
6.   * @date Anno Accademico 2018/19.
7. */
8. #ifndef _PILA_H_INCLUDED
9. #define _PILA_H_INCLUDED
10. #include<assert.h>
11. #include "Cella_Pila_P_Mono.h"
12.
13.
14. /**
15.  * @brief Pila
16.  * Questa classe contiene tutti gli operatori tipici della struttura dati "Pila".
17.  * @note Pila p=< e1,..., en>
18.  * @author Sconosciuto, modificato da Niccolo' Petti.
19. */
20. template <class T> class Pila {
21.
22. public:
23.     typedef T tipoelem;
24.
25.     Pila();
26.     ~Pila();
27. /**
28.  * @brief Crea una pila vuota.
29.  * @post la nuova pila p e' la pila vuota (p=<>).
30. */
31.     void creapila();
32.
33. /**
34.  * @brief Controlla se la pila e' vuota.
35.  * @return true se la pila e' vuota (p=<>), false altrimenti.
36. */
37.     bool pilavuota() const;
38.
39. /**
40.  * @brief Restituisce il valore dell'ultimo elemento inserito,
41.  * senza estrarlo dalla pila
42.  * @pre la pila non deve essere vuota (p!=<>)
43.  * @return l'ultimo elemento inserito (e1)
44. */
45.     tipoelem leggipila() const;
46.
47. /**
48.  * @brief Estraе dalla pila l'ultimo elemento inserito
49.  * @pre la pila non deve essere vuota (p!=<>)
50.  * @post L'ultimo elemento inserito e' rimosso dalla pila (p=<e2,...,en>)
51. */
52.     void fuoripila();
53.
54. /**
55.  * @brief Inserisce un nuovo elemento nella pila
56.  * @param[in] elem L'elemento da inserire
57.  * @post l'elemento @a elem e' inserito nella pila (p=<elem,e1,...,en>)
58. */
59.     void inpila(tipoelem elem);
60.
61. private:
62.     typedef Cella_Pila_P_Mono<T> posizione;
63.     posizione* testa;
64. };
65.
66.
67. template <class T> Pila<T>::Pila() {
68.     creapila();
69. }
70.
71. template <class T> Pila<T>::~Pila() {

```

```

72.     while(!pilavuota())
73.         fuoripila();
74.         delete testa;
75.     }
76.
77.     template <class T> void Pila<T>::creapila() {
78.         testa = nullptr;
79.     }
80.
81.     template <class T> bool Pila<T>::pilavuota() const{
82.         return(testa == nullptr);
83.     }
84.
85.     template <class T> T Pila<T>::leggipila() const{
86.         assert(!pilavuota());
87.         return testa->leggiCella();
88.     }
89.
90.     template <class T> void Pila<T>::fuoripila(){
91.         assert(!pilavuota());
92.         posizione* temp = new posizione;
93.         temp=testa;
94.         testa=testa->leggiSucc();
95.         delete temp;
96.     }
97.
98.     template <class T> void Pila<T>::inpila(tipoelem elem){
99.         posizione* temp = new posizione;
100.            temp->scriviCella(elem);
101.            temp->scriviSucc(testa);
102.            testa=temp;
103.        }
104.    #endif // _PILA_H_INCLUDED

```

6.4 CODA

Inserite tra le strutture intercambiabili le seguenti realizzazioni:

6.4.1 Coda con lista

```
7 #ifndef CODA_H_
8 #define CODA_H_
9 #include "Lista.h"
10 #include <iostream>
11
12 template <class tipoelem>
13 class Coda
14 {
15     public:
16         Coda();
17         ~Coda();
18         void creacoda();
19         bool codavuota() const;
20         void incoda(tipoelem);
21         tipoelem leggicoda() const;
22         void fuoricoda();
23
24     private:
25         Lista<tipoelem> codaL;
26 };
27
28 template<class tipoelem>
29 Coda<tipoelem>::Coda()
30 {
31     creacoda();
32 }
33
34 template<class tipoelem>
35 Coda<tipoelem>::~Coda()
36 {
37     codaL.~Lista();
38 }
```

```

41 template<class tipoelem>
42 void Coda<tipoelem>::creacoda()
43 {
44     codaL.crealista();
45 };
46
47 template<class tipoelem>
48 bool Coda<tipoelem>::codavuota()
49 {
50     return(codaL.listavuota());
51 };
52
53 template<class tipoelem>
54 void Coda<tipoelem>::incoda(tipoelem elem)
55 {
56     typename Lista<tipoelem>::posizione pos=codaL.primolista();
57
58     while(!codaL.finellista(pos))
59     {
60         pos=codaL.succlista(pos);
61     }
62
63     codaL.inslista(elem, pos);
64 };
65
66 template<class tipoelem>
67 tipoelem Coda<tipoelem>::leggicoda()
68 {
69     if(!codavuota())
70         return(codaL.leggilista(codaL.primolista()));
71 };
72
73 template<class tipoelem>
74 void Coda<tipoelem>::fuoricoda()
75 {
76     typename Lista<tipoelem>::posizione pos=codaL.primolista();
77     if(!codavuota())
78         codaL.canclista(pos);
79 };
80
81 #endif

```

6.4.2 Coda con puntatori

```
/*
 Coda.h

 Created by: Mirco Sternativo
 Edited by: Matteo Luceri

 Date: 16-Mar-2015 // 30-Oct-2019
 */

#ifndef _CODA_H
#define _CODA_H

#include "Cella.h"
#include <iostream>
#include <cstdlib>
using namespace std;

template<class tipoelem>
class Coda{

public:

    //COSTRUTTORE E DISTRUTTORE -----
    Coda();                                //costruttore coda
    Coda(Coda&);                          //costruttore di copia
    ~Coda();                               //distruttore coda

    //OPERATORI -----
    void creacoda();
    bool codavuota() const;
    tipoelem leggicoda() const;
    void incoda(tipoelem);
    void fuoricoda();
    //NUOVI METODI
    void inverti_coda();
    void svuota();

private:
    //DEFINIZIONE DEI TIPI -----
    typedef Cella<tipoelem>* posizione;
    //VARIABILI -----
    posizione testa;
    posizione fondo;

};

}
```

```

//IMPLEMENTAZIONI COSTRUTTORE E DISTRUTTORE -----
-----
template<class tipoelem>
Coda<tipoelem>::Coda(){
    creacoda();
}

template<class tipoelem>
Coda<tipoelem>::Coda(Coda<tipoelem>& c){
    //N.B.: questo costruttore effettua una copia o clone di un oggetto
    creacoda();
    tipoelem temp;
    Coda<tipoelem> comodo;

    while (!c.codavuota()){
        comodo.incoda(c.leggicoda());
        //copio gli elementi di c in una coda d'appoggio
        c.fuoricoda();                                //distruzione della coda
    }

    while (!comodo.codavuota()){
        temp=comodo.leggicoda();
        comodo.fuoricoda();
        incoda(temp);
        //copia degli elementi della coda d'appoggio nella nuova coda
        c.incoda(temp);
        //e ripristino c
    }
}
}

template<class tipoelem>
Coda<tipoelem>::~Coda(){
    while (!codavuota())
    {
        fuoricoda();
        //eliminazione elementi coda
    }
    delete fondo;
    //eliminazione riferimenti per inizio e fine coda
    delete testa;
}
}

//IMPLEMENTAZIONI OPERATORI -----
-----
template<class tipoelem>
void Coda<tipoelem>::creacoda(){
    testa=nullptr;
    fondo=nullptr;
}

template<class tipoelem>
bool Coda<tipoelem>::codavuota() const{
    return ((testa==nullptr));
    //controllo esistenza coda tramite verifica puntatori inizio e fine
}

```

```

template<class tipoelem>
tipoelem Coda<tipoelem>::leggicoda() const {
    if (!codavuota())
        //precondizione coda non vuota
        return (testa->leggicella());
    //lettura elemento in testa
}

```

```

template<class tipoelem>
void Coda<tipoelem>::incoda(tipoelem a){
    posizione temp=new Cella<tipoelem>;
    //creazione nuovo elemento temp
    temp->scrivicella(a);
    //valorizzazione di temp
    temp->scrivisucc(nullptr);

    if (!codavuota())
        fondo->scrivisucc(temp);
        fondo=temp;
    //se la coda non è vuota, l'elemento successivo è temp
    else
        testa=temp;
        fondo=temp;
    //altrimenti sta alla testa
}


```

```

template<class tipoelem>
void Coda<tipoelem>::fuoricoda(){
    if (!codavuota()){
        //precondizione coda non vuota

        posizione temp=testa;
        //puntatore all'elemento da eliminare

        testa=testa->leggisucc();
        delete temp;
        //eliminazione elemento in testa
    }
}

// NUOVI METODI-----
-----
```

```

template<class tipoelem>
void Coda<tipoelem>::inverti_coda(){
    tipoelem Elemento;

    if (!codavuota()){
        Elemento = leggicoda();
        fuoricoda();
        inverti_coda();
        incoda(Elemento);
    }
}
```

```
template<class C>
```

```
void Coda<C>::svuota(){
    while (!codavuota()){
        fuoricoda();
    }
};

#endif // CODA_H
```

6.4.3 Coda con Vettore sequenziale

```
#include <iostream>

#pragma once

#define NMAX 1024

template<class T>

class Coda {

public:

    Coda();
    ~Coda();

    void creacoda();
    bool codavuota();
    void fuoricoda();
    T leggicoda();
    void incoda(T);

private:

    typedef T tipoelem;
    typedef int posizione;
    tipoelem vett[NMAX];
    int testa;
    int fondo;
};

template<class tipoelem>Coda<tipoelem>::Coda() {
    creacoda();
}

template<class tipoelem>
Coda<tipoelem>::~Coda() {
    while(!codavuota())
    {
        fuoricoda();
    }
}
```

```

template<class tipoelem>void Coda<tipoelem>::creacoda() {
    fondo = 0;
    testa = 0;
}

template<class tipoelem>bool Coda<tipoelem>::codavuota() {
    return (fondo == 0);
}

template<class tipoelem>void Coda<tipoelem>::fuoricoda() {
    if (!codavuota()) {
        int i;
        for (i = testa; i < fondo - 1; i++) {
            vett[i] = vett[i + 1];
        }
        fondo--;
    } else {
        std::cout << "La Coda è vuota" << std::endl;  }  }
}

template<class tipoelem>tipoelem Coda<tipoelem>::leggicoda() {
    return vett[testa];
}

template<class tipoelem>void Coda<tipoelem>::incoda(tipoelem a) {
    if (fondo < NMAX) {
        vett[fondo] = a;
        fondo++;
    } }

```

6.5 CODA DI PRIORITÀ

6.5.1 Coda con priorità con vettore (Heap)

```

#include "Priorielem.h"
#include <iostream>
#include <cstdlib>

```

```
using namespace std;

//realizzazione di una coda con priorità tramite una lista (dinamica monodirezionale)
non ordinata

//valore di priorità minore indica priorità maggiore

const int MAX=1024;

template<class P> class Prioricoda

{

public:

//definizione di tipo

typedef Priorielem<P> tipoelem;

//costruttori

Prioricoda();

Prioricoda(const Prioricoda<P>&);

//distruttore di default

//operatori di specifica

void creaprioricoda();

void inserisci(tipoelem);

tipoelem min() const;

void cancellamin();

bool vuota()const;

private:
```

```

tipoelem prioricoda[MAX]; //la coda con priorità di fatto è una lista
int ultimo;

friend ostream& operator<< (ostream& o, const Prioricoda<P>& p)
//sovraffunzione output

{
    o<<p.prioricoda;
    return o;
}

};


```

```

template<class P> Prioricoda<P>::Prioricoda() //costruttore generico
{
    creaprioricoda();
}


```

```

template <class P> Prioricoda<P>::Prioricoda(const Prioricoda<P>& p)
//costruttore di copia

{
    creaprioricoda();
    prioricoda=p.prioricoda;
}


```

```

template<class P> void Prioricoda<P>::creaprioricoda() //crea la coda con
priorità

{
    ultimo=0;
}


```

```
template<class P> bool Prioricoda<P>::vuota()const //crea la coda con priorità
{
    return (ultimo==0);
}
```

```
template<class P> void Prioricoda<P>::inserisci(tipoelem a) //inserimento
```

```
{
    int i,k;
    tipoelem temp;
    if (ultimo == MAX)
    { cout<<"Coda con priorità piena"<<endl; }
    else{
        ultimo =ultimo +1;
        prioricoda[ultimo] = a;
        i = ultimo;
        if (i>1){ k = i/2;}
        while ( i>1 && prioricoda[i]<prioricoda[k] ){
            temp = prioricoda[i];
            prioricoda[i] = prioricoda[k];
            prioricoda[k] = temp;
            i = k;
            if (i>1){ k = i/2;};
        }
    }
}
```

```

template<class P> Priorielem<P> Prioricoda<P>::min() const //restituisce il
minimo della coda
{
    if(vuota()){

        cout<<"Coda con priorità vuota"<<endl;
    }

    else{

        return prioricoda[1];
    }

}

```

template<class P> void Prioricoda<P>::cancellamin() //elimina il minimo dalla

coda

```

    {

        int i,k;
        tipoelem temp;
        bool scambio=false;

        if (vuota())
        { cout<<"Coda con priorità vuota"<<endl; }

        else{

            prioricoda[1] = prioricoda[ultimo];
            ultimo = ultimo-1;
            i = 1;
            scambio = true;
            while ( i<=(ultimo/2) && scambio){

                k = 2*i;

```

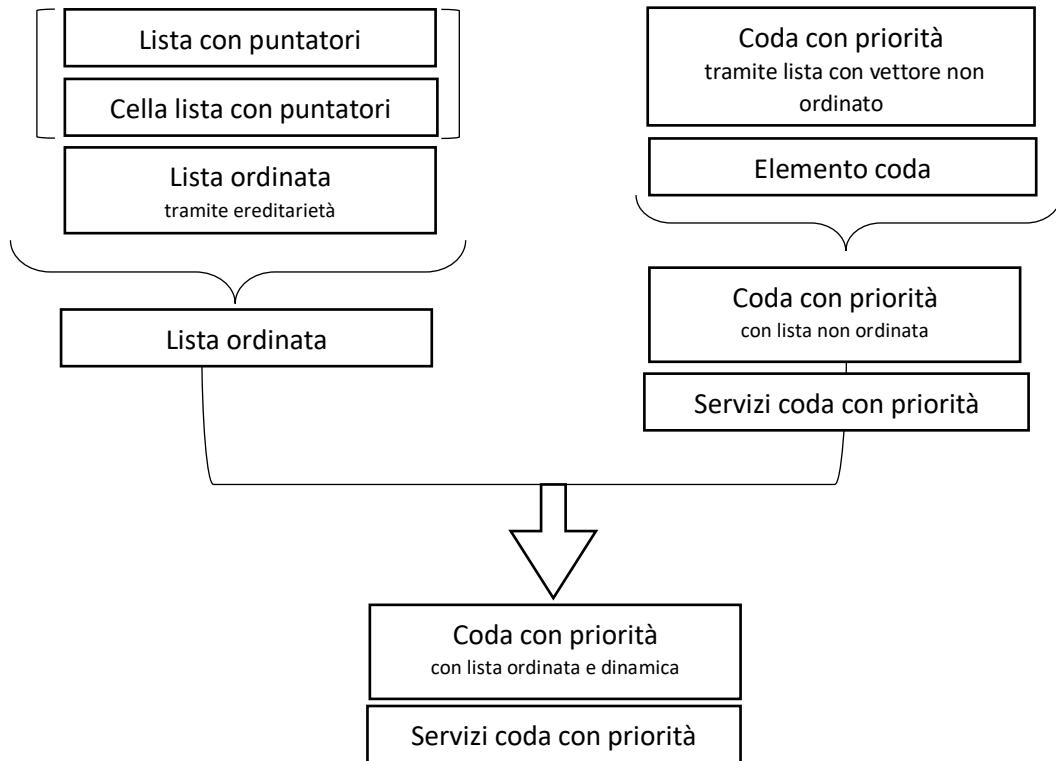
```

if (k < ultimo) {
    if (prioricoda[k] > prioricoda[k+1]){
        k=k+1;
    }
    if (prioricoda[k] < prioricoda[i]){
        temp = prioricoda[i];
        prioricoda[i] = prioricoda[k];
        prioricoda[k] = temp;
        i = k;
    }
    else{
        scambio = false;
    }
}
}

```

6.5.2 Coda con priorità con lista ordinata

Si fornisce uno schema riassuntivo del processo di creazione della struttura:



6.5.2.1 Cella_Lista_Con_Puntatori_Monodirezionale.h

```

/*
Realizzazione: Cella per Lista con Puntatori Monodirezionale
Modificato da: Matteo Luceri
Autore : Sconosciuto
*/
#ifndef CELLA_LISTA_CON_PUNTATORI_MONODIREZIONALE_H_
#define CELLA_LISTA_CON_PUNTATORI_MONODIREZIONALE_H_

template<class T>
class Cella_LP_Mono
{
public:
    typedef T tipoelem;
    Cella_LP_Mono();
    ~Cella_LP_Mono();

    void scriviCella(tipoelem);
    tipoelem leggiCella() const;
    void scriviSucc(Cella_LP_Mono<T>* );
    Cella_LP_Mono<T>* leggiSucc() const;

private:
    tipoelem elemento;
    Cella_LP_Mono<T>* succ;
};

template<class T>
Cella_LP_Mono<T>::Cella_LP_Mono()
{
    succ = nullptr;
}

template<class T>
Cella_LP_Mono<T>::~Cella_LP_Mono()

```

```

{
    //dtor
}

template<class T>
void Cella_LP_Mono<T>::scriviCella(tipoelem label)
{
    elemento = label;
}

template<class T>
T Cella_LP_Mono<T>::leggiCella() const
{
    return elemento;
}

template<class T>
void Cella_LP_Mono<T>::scriviSucc(Cella_LP_Mono<T>* c)
{
    succ = c;
}

template<class T>
Cella_LP_Mono<T>* Cella_LP_Mono<T>::leggiSucc() const
{
    return succ;
}

```

```
#endif /* CELLA_LISTA_CON_PUNTATORI_MONODIREZIONALE_H_ */
```

6.5.2.2 Lista_Con_Puntatori_Monodirezionale.h

```

/*
Realizzazione: Lista con Puntatori Monodirezionale
Modificato da Matteo Luceri
Autore : Sconosciuto
Nota Bene: Il costruttore di copia e' stato implementato in quanto
NECESSARIO per via di alcuni file del progetto in cui
vengono passate liste come parametri di funzioni e
restituite (dalle stesse) non per riferimento.
*/

```

```
#ifndef LISTA_CON_PUNTATORI_MONODIREZIONALE_H_
#define LISTA_CON_PUNTATORI_MONODIREZIONALE_H_
```

```
#include "Cella_Lista_Con_Puntatori_Monodirezionale.h"
```

```

template<class L>
class Lista
{
public:
    typedef Cella_LP_Mono<L>* posizione;
    typedef L tipoelem;

    Lista();
    Lista(const Lista&);
    ~Lista();

    void crealista();
    bool listavuota() const;
    bool finelista(posizione) const;
    posizione primolista() const;
    posizione succlista(posizione) const;
    posizione preclista(posizione) const;
    tipoelem leggilista(posizione) const;
    void scrivilista(tipoelem, posizione);
    void inslista(tipoelem, posizione&);
    void canclista(posizione &);

private:
    posizione lista;
};
```

```

template <class L>
Lista<L>::Lista()
{
    crealista();
}

template <class L>
Lista<L>::Lista(const Lista& b)
{
    crealista();
    typename Lista<L>::posizione ind = primolista(),ind2 = b.primolista();
    while (!b.finelista(ind2))
    {
        inslista(b.leggilista(ind2),ind);
        ind = succlista(ind);
        ind2 = b.succlista(ind2);
    }
}

template <class L>
Lista<L>::~Lista()
{
    posizione p = primolista();
    while(p->leggiSucc() != nullptr)
    {
        posizione temp = p;
        p=p->leggiSucc();
        delete temp;
    }
}

template <class L>
void Lista<L>::crealista()
{
    lista = new Cella_LP_Mono<L>;
    lista->scriviSucc(nullptr);
}

template <class L>
bool Lista<L>::listavuota() const
{
    return (lista->leggiSucc() == nullptr);
}

template <class L>
bool Lista<L>::finelista(posizione pos) const
{
    return (pos->leggiSucc() == nullptr);
}

template <class L>
typename Lista<L>::posizione Lista<L>::primolista() const
{
    return lista;
}

template <class L>
typename Lista<L>::posizione Lista<L>::succlista(posizione pos) const
{
    if (lista->leggiSucc() != nullptr)
        return (pos->leggiSucc());
    else
        return pos;
}

template <class L>
typename Lista<L>::posizione Lista<L>::preclista(posizione pos) const
{
    typename Lista<L>::posizione temp = primolista();
    if(pos == temp)
        temp = nullptr;
    else
    {
        while(succlista(temp) != pos)

```

```

        temp = succlista(temp);
    }
    return temp;
}

template <class L>
L Lista<L>::leggilista(posizione pos) const
{
    return (pos->leggiCella());
}

template <class L>
void Lista<L>::scrivilista(tipoelem elem, posizione pos)
{
    pos->scriviCella(elem);
}

template <class L>
void Lista<L>::inslista(tipoelem elem, posizione &pos)
{
    typename Lista<L>::posizione temp;
    temp = new Cella_LP_Mono<L>;
    temp->scriviCella(elem);
    temp->scriviSucc(pos);

    if (pos == primolista())
        lista = temp;
    else
        preclista(pos)->scriviSucc(temp);
    pos = temp;
}

template <class L>
void Lista<L>::canclista(posizione & pos)
{
    typename Lista<L>::posizione temp = pos;

    if(pos != lista)
        preclista(pos)->scriviSucc(pos->leggiSucc());
    else
        lista = pos->leggiSucc();

    pos = pos->leggiSucc();
    delete temp;
}

```

#endif /* LISTA_CON_PUNTATORI_MONODIREZIONALE_H_ */

6.5.2.3 *Lista_Oordinata_Ereditata.h*

```

/*
Definizione della struttura dati "Lista Ordinata".
Definizione di Lista Ordinata per la realizzazione tramite ereditarietà da Lista.
Autore: Andrea Esposito.
Modificata da: Matteo Luceri
*/

```

```

#ifndef LISTA_ORDINATA_EREDITATA_H_
#define LISTA_ORDINATA_EREDITATA_H_

#include "Lista_Con_Puntatori_Monodirezionale.h"

template <class T>
class ListaOrdinata : private Lista<T>
{
public:
    typedef typename Lista<T>::posizione posizione;

    ListaOrdinata();
    ListaOrdinata(const ListaOrdinata&);
    ~ListaOrdinata();

    void crealista();
    bool listavuota() const;
    bool finelista(posizione) const;

```

```

    posizione primolista() const;
    posizione succlista(posizione) const;
    posizione preclista(posizione) const;
    T leggilista(posizione) const;
    void inslista(T);
    void canclista(posizione&);
};

template <class T>
ListaOrdinata<T>::ListaOrdinata() :
    Lista<T>() // call parent constructor
{
}

template <class T>
ListaOrdinata<T>::ListaOrdinata(const ListaOrdinata<T>& l):
    Lista<T>(l) // call parent constructor
{
}

template <class T>
ListaOrdinata<T>::~ListaOrdinata()
{
    Lista<T>::~Lista();
}

template <class T>
void ListaOrdinata<T>::crealista()
{
    Lista<T>::crealista();
}

template <class T>
bool ListaOrdinata<T>::listavuota() const
{
    return Lista<T>::listavuota();
}

template <class T>
typename ListaOrdinata<T>::posizione ListaOrdinata<T>::primolista() const
{
    return Lista<T>::primolista();
}

template <class T>
bool ListaOrdinata<T>::finelista(posizione p) const
{
    return Lista<T>::finelista(p);
}

template <class T>
typename ListaOrdinata<T>::posizione ListaOrdinata<T>::succlista(posizione p) const
{
    return Lista<T>::succlista(p);
}

template <class T>
typename ListaOrdinata<T>::posizione ListaOrdinata<T>::preclista(posizione p) const
{
    return Lista<T>::preclista(p);
}

template <class T>
void ListaOrdinata<T>::inslista(T el)
{
    posizione p = primolista();
    if(!listavuota())
        while(leggilista(p) < el && !finelista(p))
            p = succlista(p);

    Lista<T>::inslista(el, p);
}

template <class T>

```

```

void ListaOrdinata<T>::canclista(posizione &p)
{
    Lista<T>::canclista(p);
}

template <class T>
T ListaOrdinata<T>::legglista(posizione p) const
{
    return Lista<T>::legglista(p);
}

#endif /* LISTA_ORDINATA_EREDITATA_H_ */

```

6.5.2.4 Elemento_Coda_Con_Priorita.h

```

/*
Realizzazione: Tipo di elemento della coda con priorità
Modificato da: Matteo Luceri
Autore :Sconosciuto
*/

#ifndef ELEMENTO_CODA_CON_PRIORITA_H_
#define ELEMENTO_CODA_CON_PRIORITA_H_

#include <iostream>
#include <cstdlib>

using namespace std;

template<class X> class Priorielem
{
public:

    //dichiarazione di tipo
    typedef float priorita; //la priorità è di tipo numerico (valore minore=priorità più alta)
    typedef X tipoelem;

    //costruttori
    Priorielem();
    Priorielem(const Priorielem& );
    Priorielem(priorita,tipoelem);

    //distruttore di default

    //setter e getter
    void scrivipriorita(priorita);
    priorita leggipriorita() const;
    void scrivielem(tipoelem);
    tipoelem leggielem() const;

    //sovraffrarchi
    void operator=(Priorielem);
    bool operator==(Priorielem);
    bool operator<(Priorielem);
    bool operator>(Priorielem);

private:
    priorita prior; //priorità
    tipoelem elem; //informazione
};

template <class X> Priorielem<X>::Priorielem() //costruttore generico
{
    prior=9999;
}

template <class X> Priorielem<X>::Priorielem(const Priorielem& p) //costruttore di copia
{
    prior=p.leggipriorita();
    elem=p.leggielem();
}

template <class X> Priorielem<X>::Priorielem(priorita p,tipoelem e) //costruttore specifico

```

```

{
    prior=p;
    elem=e;
}

template <class X> void Priorielem<X>::scrivipriorita(priorita p)
{
    prior=p;
}

template <class X> void Priorielem<X>::scrivielem(tipoelem e)
{
    elem=e;
}

template <class X> float Priorielem<X>::leggipriorita() const
{
    return(prior);
}

template <class X> X Priorielem<X>::leggielem() const
{
    return(elem);
}

//sovraffiglio
template <class X> void Priorielem<X>::operator=(Priorielem<X> p) //assegnamento
{
    prior=p.leggipriorita();
    elem=p.leggielem();
}

template <class X> bool Priorielem<X>::operator==(Priorielem<X> p) //uguaglianza
{
    return (elem==p.leggielem());
}

template <class X> bool Priorielem<X>::operator<(Priorielem<X> p) //maggioranza (solo sulla priorità)
{
    return (prior<p.leggipriorita());
}

template <class X> bool Priorielem<X>::operator>(Priorielem<X> p) //minoranza (solo sulla priorità)
{
    return (prior>p.leggipriorita());
}

//sovraffiglio output

template<class X> ostream& operator<<(ostream& os, const Priorielem<X>& p)
{
    os<<"("<<p.leggipriorita()<<"|"<<p.leggielem()<<")";
    return(os);
}

#endif /* ELEMENTO_CODA_CON_PRIORITA_H_ */

```

6.5.2.5 Coda_Con_Priorita.h

```

/*
Realizzazione: Coda con priorità tramite una lista ordinata
Note: Valore di priorità minore indica priorità maggiore
Modificato da: Matteo Luceri
Autore : Sconosciuto
*/

```

```
#ifndef CODA_CON_PRIORITA_H_
```

```

#define CODA_CON_PRIORITA_H_

#include "Lista_Ordinata_Ereditata.h"
#include <iostream>
#include <cstdlib>
#include "Elemento_Coda_Con_Priorita.h"

using namespace std;

//realizzazione di una
template<class P> class Prioricoda
{
public:

    //definizione di tipo
    typedef Priorielem<P> tipoelem;

    //costruttori
    Prioricoda();
    Prioricoda(const Prioricoda<P>&);

    //distruttore di default

    //operatori di specifica
    void creaprioricoda();
    void inserisci(tipoelem);
    tipoelem min() const;
    void cancellamin();

    //operatori ereditati dell'insieme
    bool insiemevuoto() const;
    bool appartiene(tipoelem) const;

private:
    ListaOrdinata<tipoelem> prioricoda; //la coda con priorità di fatto è una lista

    friend ostream& operator<< (ostream& o, const Prioricoda<P>& p) //sovraffaccio output
    {
        o<<p.prioricoda;
        return o;
    }
};

template<class P> Prioricoda<P>::Prioricoda() //costruttore generico
{
    creaprioricoda();
}

template <class P> Prioricoda<P>::Prioricoda(const Prioricoda<P>& p) //costruttore di copia
{
    creaprioricoda();
    prioricoda=p.prioricoda;
}

template<class P> void Prioricoda<P>::creaprioricoda() //crea la coda con priorità
{
    prioricoda.crealista();
}

template<class P> void Prioricoda<P>::inserisci(tipoelem a) //inserimento
{
    prioricoda.inslista(a);
}

template<class P> Priorielem<P> Prioricoda<P>::min() const //restituisce il minimo della coda
{
    tipoelem m;
    if (!prioricoda.listavuota()) //precondizione coda non vuota
    {
        typename ListaOrdinata<tipoelem>::posizione indice=prioricoda.primolista();
        m= prioricoda.leggilista(indice);
    }

    return m;
}

```

```

}

template<class P> void Prioricoda<P>::cancellamin() //elimina il minimo dalla coda
{
    if (!prioricoda.listavuota()) //precondizione coda non vuota
    {
        typename ListaOrdinata<tipoelem>::posizione indice=prioricoda.primolista();
        prioricoda.canclista(indice);
    }
}

template<class P> bool Prioricoda<P>::insiemevuoto() const //verifica se la coda è vuota
{
    return (prioricoda.listavuota());
}

template<class P> bool Prioricoda<P>::appartiene(tipoelem a) const //verifica se l'elemento appartiene alla coda
{
    bool trovato=false;
    if (!prioricoda.listavuota())
    {
        typename ListaOrdinata<tipoelem>::posizione indice=prioricoda.primolista(); //ricerca
        while (!prioricoda.finlista(indice) && !trovato)
        {
            if (prioricoda.leggilista(indice)==a) //se l'elemento corrente è il cercato
                trovato=true;
            else indice=prioricoda.succlista(indice);
        }
    }
    return(trovato);
}

#endif /* CODA_CON_PRIORITA_H_ */

```

6.5.2.6 Servizi_Coda_Con_Priorita.h

```

/*
Definizione dei servizi per la struttura Coda Con Priorità
Autore: Sconosciuto
Modificata da: Matteo Luceri
*/

```

```

#ifndef SERVIZI_CODA_CON_PRIORITA_H_
#define SERVIZI_CODA_CON_PRIORITA_H_

#include <string>
#include <iostream>
#include <fstream>
#include <exception>

#include "Coda_Con_Priorita.h"

template <class T>
void stampaPrioricoda(Prioricoda<T>&);

template <class T>
void inputPrioricodaDaFile(Prioricoda<T>&, std::ifstream&);

template <class T>
void outputPrioricodaSuFile(Prioricoda<T>&, std::ofstream&);

// Implementazione

template <class T>
void stampaPrioricoda(Prioricoda<T>& p)
{
    if(!p.insiemevuoto())
    {
        typename Prioricoda<T>::tipoelem el = p.min();
        std::cout << el.leggiedata() << " - con priorità: " << el.leggipriorita() << std::endl;
    }
}

```

```

        p.cancellamin();

        stampaPrioriCoda(p);

        p.inserisci(el);
    }

template <class T>
void inputPrioriCodaDaFile(Prioricoda<T>& p, std::ifstream& file)
{
    if(!file.fail())
    {
        if(file.peek() != ifstream::traits_type::eof())
            while(!file.eof())
            {
                T content;
                string s;
                getline(file, s, ',');
                file >> content;
                typename Prioricoda<T>::tipoelem::priorita priori = (typename
Prioricoda<T>::tipoelem::priorita) atof(s.c_str());
                p.inserisci(typename Prioricoda<T>::tipoelem(priori, content));
            }
        }
        else
            throw std::runtime_error("Errore di apertura del file");
    }

template <class T>
void outputPrioriCodaSuFile(Prioricoda<T>& p, std::ofstream& file)
{
    if(!file.fail())
    {
        if(!p.insiemevuoto())
        {
            typename Prioricoda<T>::tipoelem el = p.min();
            file << el.leggipriorita() << "," << el.leggiel(<< std::endl;
            p.cancellamin();

            outputPrioriCodaSuFile(p, file);

            p.inserisci(el);
        }
    }
    else
        throw std::runtime_error("Errore di apertura del file");
}

#endif /* SERVIZI_CODA_CON_PRIORITA_H_ */

```

6.6 ALBERO N-ARIO

6.6.1 AlberoNario: realizzazione con alberi binari

```

1  #ifndef ALBERONARIO_H
2  #define ALBERONARIO_H
3  #include <exception>
4  #include "BinAlbero.h"
5  /**
6  * Realizzazione dell'albero ennario attraverso l'albero binario.
7  * La memorizzazione dell'albero avviene nel seguente modo:

```

```

8     * 1. il PRIMOFIGLIO sarà FIGLIO SINISTRO.
9     * 2. il SUCCFRATELLO sarà FIGLIO DESTRO.
10    *
11    * author: Regina Zaccaria.
12    */
13    template <class tipoelem>
14    class Albero{
15
16    public:
17        //DEFINIZIONE DI NODO
18        typedef typename BinAlbero<tipoelem>::Nodo nodo;
19        //DEFINIZIONE DI NODO NULLO
20        static constexpr typename BinAlbero<tipoelem>::Nodo nil =BinAlbero<tipoelem>::nil;
21
22        //COSTRUTTORE
23        Albero();
24        //COSTRUTTORE DI COPIA
25        Albero(const Albero&);
26        //DISTRUTTORE
27        virtual ~Albero();
28        //METODO CHE CREA ALBERONARIO
29        void creaalbero();
30        //METODO CHE RESTITUISCE VERO SE L'ALBERO È VUOTO, FALSO ALTRIMENTI
31        bool alberovuoto() const;
32        //METODO CHE INSERISCE UN NODO NULLO COME RADICE
33        void insradice();
34        //METODO CHE RESTITUISCE IL NODO DELLA RADICE
35        nodo radice() const;
36        //METODO CHE RESTITUISCE IL NODO PADRE DEL NODO PASSATO
37        nodo padre(nodo);
38        //METODO CHE RESTITUISCE VERO SE IL NODO PASSATO È FOGLIA (QUINDI NON HA FIGLI), FA
39        //LSO ALTRIMENTI
40        bool foglia(nodo) const;
41        //METODO CHE RESTITUISCE IL NODO PRIMOFIGLIO DI UN NODO PASSATO
42        nodo primofratello(nodo) const;
43        //METODO CHE RESTITUISCE VERO SE IL NODO PASSATO NON HA FRATELLI SUCCESSIVI, FALSO
44        //ALTRIMENTI
45        bool ultimofratello(nodo) const;
46        //METODO CHE RESTITUISCE IL NODO FRATELLO SUCCESSIVO DEL NODO PASSATO
47        nodo succfratello(nodo)const;
48        //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO C
49        //OME NODO PRIMOFIGLIO DEL NODO PASSATO
50        void insprimosottoalbero(nodo, Albero&);
51        //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO C
52        //OME UN NODO FRATELLO SUCCESSIVO DEL NODO PASSATO
53        void inssottoalbero(nodo, Albero&);
54        //METODO CHE PASSATO UN NODO CANCELLA LO STESSO E TUTTI I SUOI DISCENDENTI
55        void cancsottoalbero(nodo);
56        //METODO CHE SCRIVE L'ETICHETTA DI TIPO TIPOELEM ALL'INTERNO DEL NODO PASSATO
57        void scrivinodo(nodo&, tipoelem);
58        //METODO CHE RESTITUISCE L'ETICHETTA DEL NODO PASSATO
59        tipoelem legginodo(nodo) const;
60
61    private:
62
63        BinAlbero<tipoelem> tree;
64        //METODO CHE COPIA UN ALBERO PASSATO PARTENDO DAL NODO SINISTRO
65        void copia_albero_sx(const BinAlbero<tipoelem>&, const nodo& ,nodo);
66        //METODO CHE COPIA L'ALBERO PASSATO PARTENDO DAL NODO DESTRO
67        void copia_albero_dx(const BinAlbero<tipoelem>&, const nodo& ,nodo);
68    };
69
70
71    template <class tipoelem>
72    Albero<tipoelem>::Albero(){
73        creaalbero();
74    }

```

```

70
71     template <class tipoelem>
72     Albero<tipoelem>::Albero(const Albero& a){
73         if(!a.tree.binalberovuoto())
74         {
75             tree.insbinradice();
76             nodo n = tree.binradice();
77             nodo secondo_albero = a.tree.binradice();
78             tree.scrivinodo(n, a.tree.legginodo(secondo_albero));
79
80             if(a.tree.sinistruvoto(a.tree.binradice()))
81             {
82                 if(!a.tree.destrovuoto(a.tree.binradice()))
83                     copia_albero_dx(a.tree,a.tree.figliodestro(a.tree.binradice()),tree
84 .binradice());
85                 else
86                     copia_albero_sx(a.tree,a.tree.figliosinistro(a.tree.binradice()),tree.binra
87 dice());
88             }
89         }
90
91     template <class tipoelem>
92     Albero<tipoelem>::~Albero(){
93         //RICHIAAMA AUTOMATICAMENTE IL DISTRUTTORE
94         //DELL'ALBERO BINARIO
95     }
96
97     template <class tipoelem>
98     void Albero<tipoelem>::creaalbero(){
99         //RICHIAAMA AUTOMATICAMENTE IL COSTRUTTORE
100        //DELL'ALBERO BINARIO
101    }
102
103    template <class tipoelem>
104    bool Albero<tipoelem>::alberovuoto() const{
105        return tree.binalberovuoto();
106    }
107
108    template <class tipoelem>
109    void Albero<tipoelem>::insradice(){
110        tree.insbinradice();
111    }
112
113    template <class tipoelem>
114    typename Albero<tipoelem>::nodo Albero<tipoelem>::radice() const{
115        return tree.binradice();
116    }
117
118    template <class tipoelem>
119    typename Albero<tipoelem>::nodo Albero<tipoelem>::padre(typename Albero<tipoelem>::nodo
120 n){
121        return tree.binpadre(n);
122    }
123
124    template <class tipoelem>
125    bool Albero<tipoelem>::foglia(typename Albero<tipoelem>::nodo n) const{
126        if(tree.sinistruvoto(n))
127            return true;
128        else
129            return false;
130    }
131
132    template <class tipoelem>
133    typename Albero<tipoelem>::nodo Albero<tipoelem>::primofiglio(typename Albero<tipoelem>
134 ::nodo n) const{

```

```

133     return tree.figliosinistro(n);
134 }
135
136 template <class tipoelem>
137 bool Albero<tipoelem>::ultimofratello(typename Albero<tipoelem>::nodo n) const{
138     if(tree.destruoto(n))
139         return true;
140     else
141         return false;
142 }
143
144 template <class tipoelem>
145 typename Albero<tipoelem>::nodo Albero<tipoelem>::succfratello(typename Albero<tipoelem>
146 >::nodo n) const{
147     return tree.figliodestro(n);
148 }
149
150 template<class tipoelem>
151 void Albero<tipoelem>::copia_albero_sx(const BinAlbero<tipoelem>& other, const nodo& ra-
152 dice_sottoalbero, nodo nodo_padre)
153 {
154     tree.insfigliosinistro(nodo_padre);
155     nodo sx = tree.figliosinistro(nodo_padre);
156     tree.scrivinodo(sx, other.legginodo(radice_sottoalbero));
157
158     if (!other.sinistruvuto(radice_sottoalbero))
159         copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), sx);
160     if (!other.destruoto(radice_sottoalbero))
161         copia_albero_dx(other, other.figliodestro(radice_sottoalbero), sx);
162 }
163
164 template <class tipoelem>
165 void Albero<tipoelem>::copia_albero_dx(const BinAlbero<tipoelem>& other, const nodo& ra-
166 dice_sottoalbero, nodo nodo_padre)
167 {
168     tree.insfigliodestro(nodo_padre);
169     nodo dx = tree.figliodestro(nodo_padre);
170     tree.scrivinodo(dx, other.legginodo(radice_sottoalbero));
171
172     if (!other.sinistruvuto(radice_sottoalbero))
173         copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), dx);
174     if (!other.destruoto(radice_sottoalbero))
175         copia_albero_dx(other, other.figliodestro(radice_sottoalbero), dx);
176 }
177
178 template <class tipoelem>
179 void Albero<tipoelem>::insprimosottoalbero(typename Albero<tipoelem>::nodo n, Albero& a)
180 {
181     nodo temp;
182     if(tree.sinistruvuto(n)){
183         copia_albero_sx(a.tree,a.tree.binradice(),n);
184     }
185     else{
186         Albero<tipoelem> T;
187         T.insradice();
188         temp=T.radice();
189         T.scrivinodo(temp,tree.legginodo(tree.figliosinistro(n)));
190
191         if(tree.sinistruvuto(tree.figliosinistro(n))){
192             if (!tree.destruoto(tree.figliosinistro(n))){
193                 copia_albero_dx(tree,tree.figliodestro(tree.figliosinistro(n)),temp
194             );
195             }
196         }
197         else{
198             copia_albero_sx(tree,tree.figliosinistro(tree.figliosinistro(n)),temp);
199         }
200     }
201 }
```

```

195     tree.cancsottobinalbero(tree.figliosinistro(n));
196
197     copia_albero_sx(a.tree,a.tree.binradice(),n);
198     temp=tree.figliosinistro(n);
199     copia_albero_dx(T.tree,T.radice(),temp);
200 }
201
202 }
203
204
205 template <class tipoelem>
206 void Albero<tipoelem>::inssottoalbero(typename Albero<tipoelem>::nodo n, Albero& a){
207     /* L'albero è ottenuto aggiungendo il sottoalbero a di radice r dove r diventa il f
208     ratello successivo
209     di n. n non è la radice*/
210
211     nodo temp;
212     if(n!=tree.binradice()){
213         if(tree.destruvoto(n)){
214             copia_albero_dx(a.tree,a.radice(),n);
215         }
216         else{
217             Albero<tipoelem> T;
218             T.insradice();
219
220             temp=T.radice();
221             T.scrivinodo(temp,tree.legginodo(tree.figliodestro(n)));
222
223             if(tree.sinistruvoto(tree.figliodestro(n))){
224                 if(!tree.destruvoto(tree.figliodestro(n))){
225                     copia_albero_dx(tree,tree.figliodestro(tree.figliodestro(n)
226                     ),temp);
227                 }
228                 else{
229                     copia_albero_sx(tree,tree.figliosinistro(tree.figliodestro(n)),temp
230                     );
231                 }
232
233                 cancsottoalbero(tree.figliodestro(n));
234
235                 copia_albero_dx(a.tree,a.radice(),n);
236                 temp=tree.figliodestro(n);
237                 copia_albero_dx(T.tree,T.radice(),temp);
238             }
239         }
240     }
241
242     template <class tipoelem>
243     void Albero<tipoelem>::cancsottoalbero(typename Albero<tipoelem>::nodo n){
244         /* L'albero è ottenuto togliendo il sottoalbero di radice n e tutti i suoi discende
245         nti*/
246         tree.cancsottobinalbero(n);
247     }
248
249     template <class tipoelem>
250     void Albero<tipoelem>::scrivinodo(typename Albero<tipoelem>::nodo& n, tipoelem elem){
251         tree.scrivinodo(n,elem);
252     }
253
254     template <class tipoelem>
255     tipoelem Albero<tipoelem>::legginodo(typename Albero<tipoelem>::nodo n) const{
256         return tree.legginodo(n);
257     }

```

6.6.1.1 BinAlbero: realizzazione totalmente dinamico

```

1. #ifndef BINALBERI_H_INCLUDED
2. #define BINALBERI_H_INCLUDED
3. #include "Nodo_Albero_Binario.h"
4.
5. /**
6. * Realizzazione dell'ALBERO BINARIO totalmente dinamica.
7. * Riferimento al padre, al figlio sinistro e al figlio destro di un nodo.
8. * author: Regina Zaccaria.
9. */
10. template <class TIPOETICHETTA>
11. class BinAlbero{
12.
13.     public:
14.         //DEFINIZIONE DEL NODO
15.         typedef Cella_Binalbero<TIPOETICHETTA>* Nodo;
16.         //DICHIARAZIONE NODO NULLO
17.         static constexpr Nodo nil=nullptr;
18.         //COSTRUTTORE
19.         BinAlbero();
20.         //DISTRUTTORE
21.         ~BinAlbero();
22.         //METODO CHE CREA UN ALBERO BINARIO
23.         void creabinalbero();
24.         //METODO CHE RESTITUISCE VERO SE L'ABERO BINARIO È VUOTO, FALSO ALTRIMENTI
25.         bool binalberovuoto()const;
26.         //METODO CHE RESTITUISCE LA RADICE DELL'ALBERO BINARIO
27.         Nodo binradice()const;
28.         //METODO CHE PASSATO UN NODO RESTITUISCE IL SUO PADRE
29.         Nodo binpadre(Nodo)const;
30.         //METODO CHE RESTITUISCE VERO SE IL FIGLIO SINISTRO DEL NODO PASSATO NON ESISTE,
31.         FALSO ALTRIMENTI
32.         bool sinistruvoto(Nodo)const;
33.         //METODO CHE RESTITUISCE VERO SE IL FIGLIO DESTRO DEL NODO PASSATO NON ESISTE, F
34.         ALSO ALTRIMENTI
35.         bool destrovuoto(Nodo)const;
36.         //METODO CHE RESTITUISCE IL FIGLIO SINISTRO DEL NODO PASSATO
37.         Nodo figliosinistro(Nodo)const;
38.         //METODO CHE RESTITUISCE IL FIGLIO DESTRO DEL NODO PASSATO
39.         Nodo figliodestro(Nodo)const;
40.         //METODO CHE DATI DUE ALBERI, CREA UNA RADICE NULLA ALL'ALBERO BINARIO IMPLICITO
41.         ,
42.         //INSERISCE COME FIGLIO SINISTRO LA RADICE DEL PRIMO ALBERO PASSATO E COPIATI I
43.         SUOI DISCENDENTI,
44.         //E INSERISCE COME FIGLIO DESTRO IL SECONDO ALBERO PASSATO E COPIATI I SUOI DISC
45.         ENDENTI.
46.         void costrbinalbero(BinAlbero<TIPOETICHETTA>&,BinAlbero<TIPOETICHETTA>&);
47.         //METODO CHE PASSATO UN NODO LO CANCELLA E CANCELLA TUTTI I SUOI DISCENDENTI
48.         void cancottobinalbero(Nodo);
49.         //METODO CHE PASSATO UN NODO RESTITUISCE LA SUA ETICHETTA
50.         TIPOETICHETTA legginodo(Nodo)const;
51.         //METODO CHE PASSATO UN NODO SCRIVE AL SUO INTERNO L'ETICHETTA DI TIPO TIPOELEM
52.         void scrivinodo(Nodo, TIPOETICHETTA);
53.         //METODO CHE INSERISCE LA RADICE NULLA
54.         void insbinradice();
55.         //METODO CHE INSERISCE COME FIGLIO SINISTRO IL NODO PASSATO
56.         void insfigliosinistro(Nodo);
57.         //METODO CHE INSERISCE COME FIGLIO DESTRO IL NODO PASSATO
58.         void insfigliodestro(Nodo);

```

```

54.
55.     private:
56.         Nodo radice;
57.     };
58.
59.
60. template <class TIPOETICHETTA>
61. BinAlbero<TIPOETICHETTA>::BinAlbero(){
62.     creabinalbero();
63. }
64.
65. template <class TIPOETICHETTA>
66. BinAlbero<TIPOETICHETTA>::~BinAlbero(){
67.     /* if(radice!=nil)
68.         cansottobinalbero(binradice());
69.     */
70.
71. }
72.
73. template <class TIPOETICHETTA>
74. void BinAlbero<TIPOETICHETTA>::creabinalbero(){
75.     radice=nil;
76. }
77.
78. template <class TIPOETICHETTA>
79.     bool BinAlbero<TIPOETICHETTA>::binalberovuoto()const{
80.         if(radice==nil)
81.             return true;
82.         else
83.             return false;
84.     }
85.
86. template <class TIPOETICHETTA>
87.     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binradice()const{
88.
89.     return radice;
90. }
91.
92. template <class TIPOETICHETTA>
93.     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binpadre(Nodo pad
94. re)const{
95.     return padre->getPadre();
96.
97. template <class TIPOETICHETTA>
98.     bool BinAlbero<TIPOETICHETTA>::sinistruvoto(Nodo sx)const{
99.         if(sx->getFiglioSx()==nil)
100.            return true;
101.        else
102.            return false;
103.    }
104.
105.   template <class TIPOETICHETTA>
106.     bool BinAlbero<TIPOETICHETTA>::destruvoto(Nodo dx)const{
107.         if(dx->getFiglioDx()==nil)
108.             return true;
109.         else
110.             return false;
111.    }
112.
113.   template <class TIPOETICHETTA>
114.     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliosini
115. stro(Nodo sx)const{
116.     return sx->getFiglioSx();
117. }
```

```

118.         typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliodest
119.     ro(Nodo dx) const{
120.         return dx->getFiglioDx();
121.     }
122.     template <class TIPOETICHETTA>
123.         void BinAlbero<TIPOETICHETTA>::costrbinalbero(BinAlbero<TIPOETICHETTA> &A,Bi
124.             nAlbero<TIPOETICHETTA> &B){
125.                 radice=new Nodo;
126.                 radice->setPadre(nil);
127.                 if(!A.binalberovuoto()){
128.                     radice->setFiglioSx(A.binradice());
129.                     radice->getFiglioSx()->setPadre(radice);
130.                 }
131.                 else
132.                     radice->setFiglioSx(nil);
133.                 if(!B.binalberovuoto()){
134.                     radice->setFiglioDx(B.binradice());
135.                     radice->getFiglioDx()->setPadre(radice);
136.                 }
137.                 else
138.                     radice->setFiglioDx(nil);
139.             }
140.         }
141.     }
142.     template <class TIPOETICHETTA>
143.         void BinAlbero<TIPOETICHETTA>::cancsottobinalbero(Nodo r){
144.             if(!sinistruvuto(r))
145.                 cancsottobinalbero(r->getFiglioSx());
146.             if(!destruvuto(r))
147.                 cancsottobinalbero(r->getFiglioDx());
148.             if(radice!=r)
149.             {
150.                 Nodo temp;
151.                 temp=binpadre(r);
152.                 if(temp->getFiglioSx()==r)
153.                     temp->setFiglioSx(nil);
154.                 else
155.                     temp->setFiglioDx(nil);
156.             }
157.             else
158.                 radice=nil;
159.             delete r;
160.         }
161.     }
162.     template <class TIPOETICHETTA>
163.         TIPOETICHETTA BinAlbero<TIPOETICHETTA>::legginode(Nodo n) const{
164.             return n->getEtichetta();
165.         }
166.     template <class TIPOETICHETTA>
167.         void BinAlbero<TIPOETICHETTA>::scrivinodo(Nodo n, TIPOETICHETTA e){
168.             n->setEtichetta(e);
169.         }
170.     template <class TIPOETICHETTA>
171.         void BinAlbero<TIPOETICHETTA>::insbinradice(){
172.             radice= new Cella_Binalbero<TIPOETICHETTA>;
173.             radice->setFiglioDx(nil);

```

```

183.         radice->setFiglioSx(nil);
184.         radice->setPadre(nil);
185.     }
186.
187.     template <class TIPOETICHETTA>
188.     void BinAlbero<TIPOETICHETTA>::insfigliosinistro(Nodo n){
189.         Nodo temp = new Cella_Binalbero<TIPOETICHETTA>;
190.         n->setFiglioSx(temp);
191.         temp->setPadre(n);
192.         temp->setFiglioDx(nil);
193.         temp->setFiglioSx(nil);
194.     }
195.
196.     template <class TIPOETICHETTA>
197.     void BinAlbero<TIPOETICHETTA>::insfigliodestro(Nodo n){
198.         Nodo temp = new Cella_Binalbero<TIPOETICHETTA>;
199.         n->setFiglioDx(temp);
200.         temp->setPadre(n);
201.         temp->setFiglioDx(nil);
202.         temp->setFiglioSx(nil);
203.     }
204.
205. #endif // BINALBERI_H_INCLUDED

```

6.6.1.2 Nodo_Albero_Binario

```

1 #ifndef NODO_ALBERO_BINARIO_H_INCLUDED
2 #define NODO_ALBERO_BINARIO_H_INCLUDED
3
4 /**
5 * Realizzazione del Nodo dell'albero binario.
6 * Il nodo conterrà riferimento al FIGLIO SINISTRO,
7 * riferimento al FIGLIO DESTRO e riferimento al PADRE.
8 * Inoltre avrà un campo ETICHETTA di tipo TIPOETICHETTA.
9 */
10 template <class TIPOETICHETTA>
11 class Cella_Binalbero{
12 public:
13     //COSTRUTTORE
14     Cella_Binalbero();
15     //DISTRUTTORE
16     ~Cella_Binalbero();
17     //METODO CHE SETTA IL FIGLIO SINISTRO
18     void setFiglioSx(Cella_Binalbero<TIPOETICHETTA*>*);
19     //METODO CHE SETTA IL FIGLIO DESTRO
20     void setFiglioDx(Cella_Binalbero<TIPOETICHETTA*>*);
21     //METODO CHE SETTA IL PADRE
22     void setPadre(Cella_Binalbero<TIPOETICHETTA*>*);
23     //METODO CHE SETTA L'ETICHETTA
24     void setEtichetta(TIPOETICHETTA&);
25     //METODO CHE RESTITUISCE IL RIFERIMENTO AL FIGLIO SINISTRO
26     Cella_Binalbero<TIPOETICHETTA*>* getFiglioSx();
27     //METODO CHE RESTITUISCE IL RIFERIMENTO AL FIGLIO DESTRO
28     Cella_Binalbero<TIPOETICHETTA*>* getFiglioDx();
29     //METODO CHE RESTITUISCE IL RIFERIMENTO AL PADRE
30     Cella_Binalbero<TIPOETICHETTA*>* getPadre();
31     //METODO CHE RESTITUISCE L'ETICHETTA
32     TIPOETICHETTA getEtichetta();
33     //OVERLOAD DELL'OPERATORE ==
34     bool operator == (Cella_Binalbero<TIPOETICHETTA>);
35     //COSTRUTTORE DI COPIA
36     Cella_Binalbero(const Cella_Binalbero& );
37     //OVERLOAD DELL'OPERATORE =
38     void operator=(const Cella_Binalbero& );
39 private:
40     Cella_Binalbero<TIPOETICHETTA*>* FiglioDx;
41     Cella_Binalbero<TIPOETICHETTA*>* FiglioSx;

```

```

42         Cella_Binalbero<TIPOETICHETTA>* Padre;
43         TIPOETICHETTA etichetta;
44     };
45
46
47     template <class TIPOETICHETTA>
48         Cella_Binalbero<TIPOETICHETTA>::Cella_Binalbero(){
49             FiglioDx=nullptr;
50             FiglioSx=nullptr;
51             Padre=nullptr;
52         }
53
54     template <class TIPOETICHETTA>
55         Cella_Binalbero<TIPOETICHETTA>::~Cella_Binalbero(){}
56
57
58     template <class TIPOETICHETTA>
59         void Cella_Binalbero<TIPOETICHETTA>::setFiglioSx(Cella_Binalbero<TIPOETICHETTA>* sx)
60     {
61         FiglioSx=sx;
62     }
63
64     template <class TIPOETICHETTA>
65         void Cella_Binalbero<TIPOETICHETTA>::setFiglioDx(Cella_Binalbero<TIPOETICHETTA>* dx)
66     {
67         FiglioDx=dx;
68     }
69
70     template <class TIPOETICHETTA>
71         void Cella_Binalbero<TIPOETICHETTA>::setPadre(Cella_Binalbero<TIPOETICHETTA>* p){
72             Padre=p;
73         }
74
75     template <class TIPOETICHETTA>
76         void Cella_Binalbero<TIPOETICHETTA>::setEtichetta(TIPOETICHETTA& e){
77             etichetta=e;
78         }
79
80     template <class TIPOETICHETTA>
81         Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getFiglioSx(){
82             return FiglioSx;
83         }
84
85     template <class TIPOETICHETTA>
86         Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getFiglioDx(){
87             return FiglioDx;
88         }
89
90     template <class TIPOETICHETTA>
91         Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getPadre(){
92             return Padre;
93         }
94
95     template <class TIPOETICHETTA>
96         TIPOETICHETTA Cella_Binalbero<TIPOETICHETTA>::getEtichetta(){
97             return etichetta;
98         }
99
100    template <class TIPOETICHETTA>
101        bool Cella_Binalbero<TIPOETICHETTA>::operator == (Cella_Binalbero<TIPOETICHETTA> b
102    ){
103        if(getEtichetta()==b.getEtichetta())
104            return true;
105        else
106            return false;
107    }
108
109    template <class TIPOETICHETTA>

```

```

106     Cella_Binalbero<TIPOETICHETTA>::Cella_Binalbero(const Cella_Binalbero& c){
107         etichetta=c.getEtichetta();
108         FiglioDx=c.getFiglioDx();
109         FiglioSx=c.getFiglioSx();
110         Padre=c.getPadre();
111     }
112
113     template <class TIPOETICHETTA>
114     void Cella_Binalbero<TIPOETICHETTA>::operator=(const Cella_Binalbero& c ){
115         etichetta=c.getEtichetta();
116         Padre=c.getPadre();
117         FiglioDx=c.getFiglioDx();
118         FiglioSx=c.getFiglioSx();
119     }
120 #endif // NODO_ALBERO_BINARIO_H_INCLUDED

```

6.6.2 AlberoNario: realizzazione primofiglio/fratello

```

1     #ifndef ALBERO_N_H
2     #define ALBERO_N_H
3     #include "Cella_albero_n.h"
4     #include <iostream>
5
6     /** Realizzazione albero ennario con primoFiglio/Fratello.
7     * Modificato da Regina Zaccaria.
8     */
9     using namespace std;
10    template <class T>
11    class Albero
12    {
13
14        public:
15            Albero(); //costruttore
16            virtual ~Albero(); //distruttore
17            static constexpr Cella<T>* nil=nullptr;
18
19            typedef T tipoelem;//definisco tipo tipoelem
20            typedef Cella<T>* nodo;//definisco tipo nodo
21            void creaalbero();//metodo che crea un albero vuoto
22            bool alberovuoto() const;//funzione che ritorna true se l'albero è vuoto
23            void insradice();//metodo che inserisce un nodo come radice dell'albero
24            nodo radice() const;//funzione che ritorna il nodo radice
25            nodo padre(nodo);//funzione che ritorna il padre di un nodo
26            bool foglia(nodo);//funzione che ritorna true se il nodo è foglia
27            nodo primofiglio(nodo);//funzione che ritorna il primofiglio di un nodo
28            bool ultimofratello(nodo);//funzione che ritorna true se il nodo non ha piu fratelli
29            nodo succfratello(nodo);//funzione che ritorna il fratello successivo di un nodo
30            void insprimosottoalbero(nodo, Albero<T> &);//metodo che inserisce un albero come primo figlio di un nodo
31            void inssottoalbero(nodo, Albero<T> &);//metodo che inserisce un albero come fratello successivo di un nodo
32            void cancottoalbero(nodo);//metodo che cancella un sotto albero
33            void scrivinodo(nodo, tipoelem);//metodo che scrive nell'etichetta del nodo
34            tipoelem legginodo(nodo);//funzione che ritorna il contenuto dell'etichetta del nodo
35        private:
36            nodo root;//nodo radice
37        };
38
39        template <class T> Albero<T>::Albero()
40        {
41            creaalbero();
42        }
43
44        template <class T> Albero<T>::~Albero() {}
45

```

```

46  template <class T> void Albero<T>::creaalbero()
47  {
48      root=NULL;
49  }
50
51  template <class T> bool Albero<T>::alberovuoto() const
52  {
53      return(this->root==NULL);
54  }
55
56  template <class T> void Albero<T>::insradice()
57  {
58      if(alberovuoto())
59      {
60          root= new Cella<T>;
61          root->setpadre(NULL);
62          root->setprimofiglio(NULL);
63          root->setfratellosucc(NULL);
64      }
65  }
66
67
68  template <class T>
69  Cella<T>* Albero<T>::radice() const{
70      return (root);
71  }
72
73  template <class T>
74  Cella<T>* Albero<T>::padre(nodo n){
75      if(n!=radice()){
76          return (n->getpadre());
77      }
78  }
79
80  template <class T>
81  bool Albero<T>::foglia(nodo n){
82      return (n->getprimofiglio()==NULL);
83  }
84
85  template <class T>
86  Cella<T>* Albero<T>::primofiglio(nodo n){
87      return (n->getprimofiglio());
88  }
89
90
91  template <class T>
92  bool Albero<T>::ultimofratello(nodo n){
93      return (n->getfratellosucc()==NULL);
94  }
95
96  template <class T>
97  Cella<T>* Albero<T>::succfratello(nodo n){
98      if(!ultimofratello(n)){
99          return (n->getfratellosucc());
100     }
101  }
102
103 template <class T>
104 void Albero<T>::insprimosottoalbero(nodo n, Albero<T> &albero){
105     if (!(albero.alberovuoto())){
106         albero.radice()->setfratellosucc(n->getprimofiglio());
107         albero.radice()->setpadre(n);
108         n->setprimofiglio(albero.radice());
109     }
110 }
111
112 template <class T>

```

```

113 void Albero<T>::inssottoalbero(nodo n, Albero<T> &albero){
114     if((!(albero.alberovuoto()) && (radice() != n)){
115         albero.radice()->setfratellosucc(n->getfratellosucc());
116         albero.radice()->setpadre(n->getpadre());
117         n->setfratellosucc(albero.radice());
118     }
119 }
120
121 template <class T>
122 void Albero<T>::cancsottoalbero(nodo n){
123     if (n==radice()){
124         delete this->root;
125     }
126     else{
127         if(n==primofiglio(padre(n))){
128             padre(n)->setprimofiglio(succfratello(n));
129         }
130         else{
131             nodo indice=primofiglio(padre(n));
132             while(succfratello(indice)!=n){
133                 indice=succfratello(indice);
134             }
135             indice->setfratellosucc(succfratello(n));
136         }
137     }
138     delete n;
139 }
140
141 template <class T>
142 void Albero<T>::scrivinodo(nodo n, tipoelem a){
143     n->setetichetta(a);
144 }
145
146 template <class T>
147 T Albero<T>::legginodo(nodo n){
148     return (n->getetichetta());
149 }
150
151
152 #endif // ALBERO_N_H

```

6.6.2.1 Cella_albero_n

```

1 #ifndef CELLA_H
2 #define CELLA_H
3 #include <iostream>
4 using namespace std;
5 template <class T>
6
7 class Cella
8 {
9 public:
10     typedef T tipoelem;
11     Cella();
12     Cella(tipoelem);
13     virtual ~Cella();
14     void setetichetta(tipoelem);
15     tipoelem getetichetta();
16     void setprimofiglio(Cell a* );
17     Cell a* getprimofiglio();
18     void setfratellosucc(Cell a* );
19     Cell a* getfratellosucc();
20     void setpadre(Cell a* );
21     Cell a* getpadre();

```

```

22     bool operator==(Cella);
23 private:
24     tipoelem etichetta;
25     Cella* primofiglio;
26     Cella* fratellosucc;
27     Cella* padre;
28 };
29
30 template <class T> Cella<T>::Cella()
31 {
32     T elemento;
33     this->etichetta=elemento;
34     this->primofiglio=NULL;
35     this->fratellosucc=NULL;
36     this->padre=NULL;
37 }
38
39 template <class T> Cella<T>::~Cella() {}
40
41 template <class T> Cella<T>::Cella(tipoelem a)
42 {
43     this->etichetta=a;
44     this->primofiglio=NULL;
45     this->fratellosucc=NULL;
46     this->padre=NULL;
47 }
48
49 template <class T> void Cella<T>::setetichetta(tipoelem a)
50 {
51     this->etichetta=a;
52 }
53
54 template <class T> T Cella<T>::getetichetta()
55 {
56     return etichetta;
57 }
58
59 template <class T> void Cella<T>::setprimofiglio(Cella* a)
60 {
61     this->primofiglio=a;
62 }
63
64 template <class T> Cella<T>* Cella<T>::getprimofiglio()
65 {
66     return primofiglio;
67 }
68
69 template <class T> void Cella<T>::setfratellosucc(Cella* a)
70 {
71     this->fratellosucc=a;
72 }
73
74 template <class T> Cella<T>* Cella<T>::getfratellosucc()
75 {
76     return fratellosucc;
77 }
78
79 template <class T> void Cella<T>::setpadre(Cella* a)
80 {
81     this->padre=a;
82 }
83
84 template <class T> Cella<T>* Cella<T>::getpadre()
85 {
86     return padre;
87 }
88

```

```

89     template <class T> bool Cella<T>::operator==(Cella a)
90     {
91         return (this->etichetta==a.getetichetta());
92     }
93
94 #endif // CELLA_H

```

6.7 INSIEMI

6.7.1 Insiemi con lista

```

1 /**
2  * @file Insieme.h
3  * Definizione della struttura dati "Insieme".
4  * @note Definizione di Insieme per la realizzazione tramite lista.
5  * @author Sconosciuto, modificato da Andrea Esposito.
6  * @date Anno Accademico 2018/19.
7 */
8 #ifndef INSIEME_H_
9 #define INSIEME_H_
10
11 #include "Lista.h"
12
13 /**
14  * @brief Insieme
15  * Questa classe contiene tutti gli operatori tipici della struttura
16  * dati "Insieme".
17  * @author Sconosciuto, modificato da Andrea Esposito.
18 */
19 template <class TipoElem>
20 class Insieme
21 {
22 public:
23     Insieme();
24     Insieme(const Insieme&);
25     ~Insieme();
26
27     /**
28      * @brief Crea un insieme vuoto.
29      * @post Il nuovo insieme i è l'insieme vuoto (i={}).
30      */
31     void creainsieme();
32
33     /**
34      * @brief Controlla se l'insieme è vuoto.
35      * @return true se l'insieme è vuoto, false altrimenti.
36      */
37     bool insiemevuoto() const;
38
39     /**
40      * @brief controlla se un elemento appartiene all'insieme.
41      * @param[in] L'elemento da controllare
42      * @return true se l'elemento appartiene all'insieme, false altrimenti.
43      */
44     bool appartiene(const TipoElem& e) const;
45
46     /**
47      * @brief Inserisce un elemento nell'insieme.
48      * @param[in] e L'elemento da aggiungere

```

```

49     * @post L'elemento e è aggiunto all'insieme.
50     */
51     void inserisci(const TipoElem& e);
52
53     /**
54      * @brief Rimuove un elemento dall'insieme.
55      * @param[in] e L'elemento da rimuovere.
56      * @post L'elemento e è rimosso dall'insieme.
57      */
58     void cancella(const TipoElem& e);
59
60     /**
61      * @brief Unione (matematica) di due insiemi.
62      * @param[in] i Insieme da unire.
63      * @pre i deve essere un insieme valido.
64      * @post L'insieme conterrà tutti e soli gli elementi
65      * iniziali e gli elementi dell'insieme i.
66      */
67     void unione(const Insieme& i);
68
69     /**
70      * @brief Intersezione (matematica) di due insiemi.
71      * @param[in] i Insieme da intersecare.
72      * @pre i deve essere un insieme valido.
73      * @post L'insieme conterrà tutti e soli gli elementi
74      * presenti sia nell'insieme iniziale che in i.
75      */
76     void intersezione(const Insieme& i);
77
78     /**
79      * @brief Differenza (matematica) di due insiemi.
80      * @param[in] i Insieme da sottrarre.
81      * @pre i deve essere un insieme valido.
82      * @post L'insieme conterrà tutti e soli gli elementi
83      * iniziali che non appartengono anche a i.
84      */
85     void differenza(const Insieme& i);
86
87     private:
88     Lista<TipoElem> setlist;
89 };
90
91     template <class TipoElem>
92     Insieme<TipoElem>::Insieme()
93     {
94         creainsieme();
95     }
96
97     template <class TipoElem>
98     Insieme<TipoElem>::Insieme(const Insieme<TipoElem>& i)
99     {
100        /*
101         * La seguente istruzione suppone che sia presente
102         * un costruttore di copia per le liste. Tale
103         * supposizione non è casuale: il C++ ne richiede
104         * uno, per cui in sua assenza ne definisce uno
105         * di default.
106         */
107        setlist = Lista<TipoElem>(i.setlist);
108    }
109
110    template <class TipoElem>
111    Insieme<TipoElem>::~Insieme()
112    {
113        typename Lista<TipoElem>::posizione p = setlist.primolista();
114        while (!setlist.finelista(p))
115            setlist.canclista(p);

```

```

116     }
117
118     template <class TipoElem>
119     void Insieme<TipoElem>::creainsieme()
120     {
121         setlist.crealista();
122     }
123
124     template <class TipoElem>
125     bool Insieme<TipoElem>::insiemevuoto() const
126     {
127         return setlist.listavuota();
128     }
129
130     template <class TipoElem>
131     bool Insieme<TipoElem>::appartiene(const TipoElem& e) const
132     {
133         bool trovato = false;
134         typename Lista<TipoElem>::posizione p = setlist.primolista();
135         while (!setlist.finelista(p) && !trovato)
136         {
137             if(setlist.legglista(p) == e)
138                 trovato = true;
139             else
140                 p = setlist.succlista(p);
141         }
142         return trovato;
143     }
144
145     template <class TipoElem>
146     void Insieme<TipoElem>::inserisci(const TipoElem& e)
147     {
148         if (!appartiene(e))
149         {
150             typename Lista<TipoElem>::posizione p = setlist.primolista();
151             setlist.inslista(e,p);
152         }
153     }
154
155     template <class TipoElem>
156     void Insieme<TipoElem>::cancella(const TipoElem& e)
157     {
158         typename Lista<TipoElem>::posizione p = setlist.primolista();
159         while (!setlist.finelista(p) && setlist.legglista(p) != e)
160             p = setlist.succlista(p);
161         if (setlist.legglista(p) == e)
162             setlist.canclista(p);
163     }
164
165     template <class TipoElem>
166     void Insieme<TipoElem>::unione(const Insieme<TipoElem>& i)
167     {
168         typename Lista<TipoElem>::posizione p = i.setlist.primolista();
169         while (!i.setlist.finelista(p))
170         {
171             inserisci(i.setlist.legglista(p));
172             p = i.setlist.succlista(p);
173         }
174     }
175
176     template <class TipoElem>
177     void Insieme<TipoElem>::intersezione(const Insieme<TipoElem>& i)
178     {
179         typename Lista<TipoElem>::posizione p = setlist.primolista();
180
181         while(!setlist.finelista(p))
182         {

```

```

183     if(!i.appartiene(setlist.legglista(p)))
184         cancella(setlist.legglista(p));
185     else
186         p = setlist.succlista(p);
187     }
188 }
189
190 template <class TipoElem>
191 void Insieme<TipoElem>::differenza(const Insieme<TipoElem>& i)
192 {
193     typename Lista<TipoElem>::posizione p = i.setlist.primolista();
194
195     while(!setlist.finlista(p))
196     {
197         if(i.appartiene(setlist.legglista(p)))
198             cancella(setlist.legglista(p));
199         else
200             p = setlist.succlista(p);
201     }
202 }
203
204 #endif /* INSIEME_H_ */

```

6.8 DIZIONARIO

6.8.1 Dizionario con vettore ordinato

```
7  #ifndef _DIZIONARIOL_H
8  #define _DIZIONARIOL_H
9  #include <iostream>
10 #include "Entry.h"
11 #include <cstdlib>
12 #define MAX 101
13 using namespace std;
14
15 template <typename K, class E>
16 class Dizionario
17 { // E=tipo elemento, K=tipo chiave
18     typedef Entry<K, E> entry;
19
20     public:
21
22         Dizionario();
23         ~Dizionario();
24         void crea();
25         bool appartiene(const K&);
26         void inserisci(K, E);
27         E recupera(const K&);
28         void aggiorna(const K&, const E&);
29         void cancella(const K&);
30
31         //operatori di servizio
32         unsigned int dimensione();
33         bool vuoto();
34         E stampa(int &);
35
36     private:
37
38     // const unsigned int MAX=101;
39     entry* diz;
40     int nelementi;
41
42 };
43
44 //costruttore e distruttore
45 template<typename K, typename E>
46 Dizionario<K, E>::Dizionario()
47 {
48     crea();
49 }
50
51 template<typename K, typename E>
52 Dizionario<K, E>::~Dizionario() {}
53
54
55 //operatori
56 template<typename K, typename E>
57 void Dizionario<K, E>::crea()
58 {
59     diz=new entry[MAX];
60     nelementi=0;
61 }
```

```

62
63     template<typename K, typename E>
64     bool Dizionario<K, E>::appartiene(const K &chiave)
65     {
66         bool trovato=false;
67         int p,u,med;
68
69         p=0;
70         u=nelementi-1;
71
72         while(p<=u && !trovato)
73         {
74             med=(p+u)/2;
75
76             if(diz[med].get_key()==chiave)
77             {
78                 trovato=true;
79             }
80             else if(diz[med].get_key() < chiave)
81             {
82                 p=med+1;
83             }
84             else
85             {
86                 u=med-1;
87             }
88         }
89     }
90     return trovato;
91 }

92
93     template<typename K, typename E>
94     void Dizionario<K, E>::inserisci(K chiave, E elemento)
95     {
96         int i=nelementi;
97         if(i==0)
98         {
99             diz[0].set_value(elemento);
100            diz[0].set_key(chiave);
101        }
102        else
103        {
104            //Ordinamento crescente RISPETTO ALLA CHIAVE
105            while(i>0 && diz[i-1].get_key() > chiave)
106            {
107                diz[i].set_value(diz[i-1].get_value());
108                diz[i].set_key(diz[i-1].get_key());
109                i--;
110            }
111            diz[i].set_value(elemento);
112            diz[i].set_key(chiave);
113        }
114        nelementi++;
115    }
116

```

```
117     template<typename K, typename E>
118     void Dizionario<K, E>::cancella(const K &chiave)
119     {
120         bool trovato=false;
121         if(nelementi>0)
122         {
123             for(int i=0; i<nelementi; i++)
124             {
125                 if(diz[i].get_key()==chiave)
126                 {
127                     trovato=true;
128                 }
129
130                 if(trovato)
131                 {
132                     diz[i].set_key(diz[i+1].get_key());
133                     diz[i].set_value(diz[i+1].get_value());
134                 }
135             }
136             if(trovato)
137             {
138                 nelementi--;
139             }
140         }
141     }
```

```
143
144     template<typename K, typename E>
145     E Dizionario<K, E>::recupera(const K &chiave)
146     {
147         int p,u,med;
148         p=0;
149         u=nelementi-1;
150
151         while(p<=u)
152         {
153             med=(p+u)/2;
154
155             if(diz[med].get_key()==chiave)
156             {
157                 return diz[med].get_value();
158             }
159             else if(diz[med].get_key() < chiave)
160             {
161                 p=med+1;
162             }
163             else
164             {
165                 u=med-1;
166             }
167         }
168     }
169 }
```

```

171     template<typename K, typename E>
172     void Dizionario<K, E>::aggiorna(const K &chiave, const E &elemento)
173     {
174         bool trovato=false;
175         int p,u,med;
176
177         p=0;
178         u=nElementi-1;
179
180         while(p<=u && !trovato)
181         {
182             med=(p+u)/2;
183
184             if(diz[med].get_key()==chiave)
185             {
186                 trovato=true;
187                 diz[med].set_key(chiave);
188                 diz[med].set_value(elemento);
189             }
190             else if(diz[med].get_key() < chiave)
191             {
192                 p=med+1;
193             }
194             else
195             {
196                 u=med-1;
197             }
198
199         }
200     }
201
202
203     template<typename K, typename E>
204     unsigned int Dizionario<K, E>::dimensione() {
205         return nElementi;
206     }
207
208
209     template<typename K, typename E>
210     bool Dizionario<K, E>::vuoto() {
211         return (nElementi == 0);
212     }
213
214
215     template<typename K, typename E>
216     E Dizionario<K, E>::stampa(int &j){
217
218         int i;
219         for(i=0;i<=nElementi;i++){
220             if (i==j){
221                 return diz[i].get_value();
222             }
223         }
224
225     #endif

```

6.8.2 Dizionario con lista

```

1 //Realizzazione dizionario con lista - MARCELLO PALAGIANO
2 #ifndef _DIZIONARIO_H
3 #define _DIZIONARIO_H
4 #include <iostream>
5 #include "Entry.h"
6 #include "Lista.h"
7 using namespace std;

```

```

8  template <typename K, class E>
9   class Dizionario { // E=tipo elemento, K=tipo chiave
10  public:
11      typedef Entry<K, E> entry;
12      //Costruttori
13      Dizionario();
14      //Distruttori
15      ~Dizionario();
16      //Operatori di specifica
17      void crea();
18      bool appartiene(const K&);
19      void inserisci(K, E);
20      E recupera(const K&);
21      void aggiorna(const K&, E&);
22      void cancella(const K&);
23  private:
24      Lista<Entry<K, E> > table;
25  };
26
27  template<typename K, typename E>
28  Dizionario<K, E>::Dizionario() {
29      crea();
30  }
31  template<typename K, typename E>
32  Dizionario<K, E>::~Dizionario() {}
33  template<typename K, typename E>
34  void Dizionario<K, E>::crea(){
35      table.crealista();
36  }
37  template<typename K, typename E>
38  bool Dizionario<K, E>::appartiene(const K &chiave){
39      bool trovato=false;
40      typename Lista<Entry<K, E> >::posizione pos=table.primolista();
41      while(!trovato && !table.finelista(pos)){
42          if(table.leggilista(pos).get_key() == chiave){
43              trovato=true;
44          }
45          else{
46              pos=table.succlista(pos);
47          }
48      }
49      return trovato;
50  }
51  template<typename K, typename E>
52  void Dizionario<K, E>::inserisci(K chiave, E elemento){
53      typename Lista<Entry<K, E> >::posizione pos=table.primolista();
54      Entry<K,E> elem;
55      elem.set_key(chiave);
56      elem.set_value(elemento);
57      table.inslista(elem,pos);
58  }
59  template<typename K, typename E>
60  void Dizionario<K, E>::cancella(const K &chiave){
61      typename Lista<Entry<K, E> >::posizione pos=table.primolista();
62      bool trovato=false;
63      while(!trovato && !table.finelista(pos)){
64          if(table.leggilista(pos).get_key() == chiave){
65              trovato=true;
66              table.canclista(pos);
67          }
68          else{
69              pos=table.succlista(pos);
70          }
71      }
72  }
73  template<typename K, typename E>
74  E Dizionario<K, E>::recupera(const K &chiave){
```

```

75     typename Lista<Entry<K, E>>::posizione pos=table.primolista();
76     Entry<K,E> elem;
77     E valore;
78     bool trovato = false;
79     while(!table.finlista(pos) && !trovato){
80         if(table.legglistा(pos).get_key() == chiave){
81             valore=table.legglistा(pos).get_value();
82             elem.set_value(valore);
83             trovato=true;
84         }
85         pos=table.succlista(pos);
86     }
87     return elem.get_value();
88 }
89 template<typename K, typename E>
90 void Dizionario<K, E>::aggiorna(const K &chiave, E &elemento){
91     typename Lista<Entry<K, E>>::posizione pos=table.primolista();
92     Entry<K,E> elem;
93     bool trovato = false;
94     while(!table.finlista(pos) && !trovato){
95         if(table.legglistा(pos).get_key() == chiave){
96             elem.set_key(chiave);
97             elem.set_value(elemento);
98             table.scrivilista(elem,pos);
99             trovato=true;
100        }
101        pos=table.succlista(pos);
102    }
103 }
104 #endif // _DIZIONARIO_H

```

6.9 GRAFI

6.9.1 Matrice di Adiacenza

```

1. /**
2.  * @file Grafo.h
3.  * Definizione struttura dati Grafo realizzata mediante Matrice di Adiacenza
4.  * @author Graziano Montanaro.
5.  * @date Anno Accademico 2018/19.
6. */
7.
8. #ifndef GRAFO_H_INCLUDED
9. #define GRAFO_H_INCLUDED
10.
11. #include "CellaGrafo.h"
12. #include "CellaNodo.h"
13. #include <iostream>
14. #include "Lista.h"
15. #define maxNodi 1024
16.
17.
18. template <class tipoElem, class tipoPeso>
19. class Grafo
20. {
21. public:
22.     class nodo;
23.     Grafo();
24.     ~Grafo();
25.     void creagrafo();
26.     bool grafovusto();
27.     void insarco(nodo&, nodo&);
28.     void cancarco(nodo&, nodo&);
29.     void insnodo(nodo &);

```

```

30.     void cancnodo(nodo&);
31.     Lista<nodo> adiacenti(nodo&);
32.     bool esistenodo(nodo&);
33.     bool esistearco(nodo&,nodo&);
34.     void scrivinodo(tipoElem,nodo&);
35.     void scriviarco(tipoPeso,nodo&,nodo&);
36.     tipoElem legginodo(nodo&);
37.     tipoPeso leggiarco(nodo&,nodo&);
38.
39.
40. private:
41.
42.     unsigned int nodiUsati;
43.     unsigned int primoLibero();
44.     CellaGrafo<tipoPeso> matrice[maxNodi][maxNodi];
45.     CellaNodo<tipoElem> nodi[maxNodi];
46. };
47.
48.
49. // Definizione del tipo nodo
50.
51. template <class tipoElem, class tipoPeso>
52. class Grafo<tipoElem,tipoPeso>::nodo
53. {
54. public:
55.     nodo();
56.     nodo(const nodo&);
57.     ~nodo();
58.     int legginodo();
59.     void scrivinodo(int n);
60.     bool operator == ( const nodo& );
61.     bool operator != ( const nodo& );
62.
63.
64. private:
65.     int valore;
66. };
67.
68.
69.
70. template <class tipoElem, class tipoPeso>
71. Grafo<tipoElem , tipoPeso>::Grafo()
72. {
73.     nodiUsati = 0;
74.     for(int i=0;i<maxNodi;i++)
75.     {
76.         nodi[i].setPresente(false);
77.         for(int j=0;j<maxNodi;j++)
78.             matrice[i][j].setUsato(false);
79.     }
80. }
81.
82. template <class tipoElem, class tipoPeso>
83. Grafo<tipoElem , tipoPeso>::~Grafo()
84. {
85.
86. }
87.
88. template <class tipoElem, class tipoPeso>
89. bool Grafo<tipoElem , tipoPeso>::grafovuito()
90. {
91.     return nodiUsati == 0;
92. }
93.
94. template <class tipoElem, class tipoPeso>
95. bool Grafo<tipoElem, tipoPeso>::esistenodo(nodo &n)
96. {

```

```

97.     if(n.legginodo()>=0 && n.legginodo()<nodiUsati)
98.         return nodi[n.legginodo()].getPresente();
99.     else
100.         return false;
101.    }
102.
103.    template <class tipoElem, class tipoPeso>
104.        tipoPeso Grafo<tipoElem , tipoPeso>::leggiarco(nodo& n1, nodo& n2)
105.    {
106.        if(esistearco(n1,n2))
107.            return matrice[n1.legginodo()][n2.legginodo()].getPeso();
108.
109.
110.    }
111.
112.    template <class tipoElem, class tipoPeso>
113.        void Grafo<tipoElem , tipoPeso>::insnodo(nodo & n)
114.    {
115.        if(!esistenodo(n))
116.        {
117.            if(nodiUsati<maxNodi)
118.            {
119.                unsigned int i = primoLibero();
120.                nodi[i].setPresente(true);
121.                nodi[i].setArchi(0);
122.                n.scrivinodo(i);
123.                nodiUsati++;
124.            }
125.        }
126.    }
127.
128.    template <class tipoElem, class tipoPeso>
129.        void Grafo<tipoElem , tipoPeso>::cancnodo(nodo& n)
130.    {
131.        if(esistenodo(n))
132.        {
133.            nodo* temp = new nodo;
134.            for(int i=0;i<nodiUsati;i++)
135.            {
136.                temp->scrivinodo(i);
137.                if(esistearco(*temp,n))
138.                    cancarco(*temp,n);
139.                if(esistearco(n,*temp))
140.                    cancarco(n,*temp);
141.            }
142.            nodi[n.legginodo()].setPresente(false);
143.            nodiUsati--;
144.
145.
146.        }
147.
148.    }
149.
150.    template <class tipoElem, class tipoPeso>
151.        void Grafo<tipoElem , tipoPeso>::insarco(nodo &n1, nodo &n2)
152.    {
153.        if(!esistearco(n1,n2))
154.        {
155.            matrice[n1.legginodo()][n2.legginodo()].setUsato(true);
156.            nodi[n1.legginodo()].setArchi(nodi[n1.legginodo()].getArchi()+1);
157.            nodi[n2.legginodo()].setArchi(nodi[n2.legginodo()].getArchi()+1);
158.        }
159.
160.
161.    }
162.
163.    template <class tipoElem, class tipoPeso>

```

```

164.     void Grafo<tipoElem , tipoPeso>::cancarco(nodo& n1, nodo& n2)
165.     {
166.         if(esistearco(n1,n2))
167.         {
168.             matrice[n1.legginodo()][n2.legginodo()].setUsato(false);
169.             nodi[n1.legginodo()].setArchi(nodi[n2.legginodo()].getArchi() -1);
170.             nodi[n2.legginodo()].setArchi(nodi[n2.legginodo()].getArchi() -1);
171.         }
172.     }
173. }
174.
175. template <class tipoElem, class tipoPeso>
176. void Grafo<tipoElem , tipoPeso>::scrivinodo(tipoElem e, nodo& n)
177. {
178.     if(esistenodo(n))
179.     {
180.         nodi[n.legginodo()].setEtichetta(e);
181.     }
182. }
183. }
184.
185. template <class tipoElem, class tipoPeso>
186. tipoElem Grafo<tipoElem , tipoPeso>::legginodo(nodo& n)
187. {
188.     if(esistenodo(n))
189.     {
190.         return nodi[n.legginodo()].getEtichetta();
191.     }
192. }
193.
194. template <class tipoElem, class tipoPeso>
195. void Grafo<tipoElem , tipoPeso>::scriviarco(tipoPeso p,nodo& n1, nodo& n2)
196. {
197.     if(!esistearco(n1,n2))
198.     {
199.         matrice[n1.legginodo()][n2.legginodo()].setPeso(p);
200.     }
201. }
202. }
203.
204. template <class tipoElem, class tipoPeso>
205. bool Grafo<tipoElem , tipoPeso>::esistearco(nodo& n1, nodo& n2)
206. {
207.     if(esistenodo(n1))
208.         if(esistenodo(n2))
209.             return matrice[n1.legginodo()][n2.legginodo()].getUsato();
210.
211. }
212.
213.
214. template <class tipoElem, class tipoPeso>
215. Lista<typename Grafo<tipoElem,tipoPeso>::nodo> Grafo<tipoElem , tipoPeso>::adiac
enti(nodo& n)
216. {
217.     Lista<nodo> l;
218.     typename Lista<nodo>::posizione p = l.primolista();
219.     for(int i=0;i<nodiUsati;i++)
220.     {
221.         if(matrice[n.legginodo()][i].getUsato())
222.         {
223.             nodo* temp = new nodo();
224.             temp->scrivinodo(i);
225.             l.inslista(*temp,p);
226.         }
227.     }
228. // Nodi di cui n è coda
229. /*for(int i=0;i<nodiUsati;i++)
```

```

230.         {
231.             if(matrice[i][n].getUsato())
232.                 if(!matrice[n][i].getUsato())
233.                     l.inslista(i,p);
234.         }/*
235.         return l;
236.     }
237.
238.     template <class tipoElem, class tipoPeso>
239.     unsigned int Grafo<tipoElem,tipoPeso>::primoLibero()
240.     {
241.         nodo i;
242.         i.scrivinodo(-1);
243.         bool libero=false;
244.         while(!libero)
245.         {
246.             i.scrivinodo(i.legginodo()+1);
247.             if(nodi[i.legginodo()].getPresente()==false)
248.                 libero = true;
249.         }
250.         return i.legginodo();
251.     }
252.
253. //Operatori NODO
254.
255.     template <class tipoElem, class tipoPeso>
256.     Grafo<tipoElem,tipoPeso>::nodo::nodo()
257.     {
258.         valore = -1;
259.     }
260.
261.
262.
263.     template <class tipoElem, class tipoPeso>
264.     Grafo<tipoElem,tipoPeso>::nodo::nodo(const nodo& n)
265.     {
266.         valore = n.valore;
267.     }
268.
269.
270.
271.     template <class tipoElem, class tipoPeso>
272.     Grafo<tipoElem,tipoPeso>::nodo::~nodo()
273.     {
274.
275.     }
276.
277.
278.     template <class tipoElem, class tipoPeso>
279.     int Grafo<tipoElem,tipoPeso>::nodo::legginodo()
280.     {
281.         return valore;
282.     }
283.
284.
285.     template <class tipoElem, class tipoPeso>
286.     void Grafo<tipoElem,tipoPeso>::nodo::scrivinodo(int n)
287.     {
288.         valore = n;
289.     }
290.
291.     template <class tipoElem, class tipoPeso>
292.     bool Grafo<tipoElem,tipoPeso>::nodo:: operator == ( const nodo& n)
293.     {
294.         return valore == n.valore;
295.     }
296.

```

```

297.     template <class tipoElem, class tipoPeso>
298.     bool Grafo<tipoElem,tipoPeso>::nodo:: operator != ( const nodo& n)
299.     {
300.         return valore != n.valore;
301.     }
302.
303.
304. #endif // GRAFO_H_INCLUDED

```

6.9.1.1 CellaGrafo.h

```

1. /**
2. * @file CellaGrafo.h
3. * Definizione della Cella presente nella matrice di Grafo.h ( matrice di adiacenza)
4. * @author Graziano Montanaro.
5. * @date Anno Accademico 2018/19.
6. */
7.
8.
9. #ifndef CELLAGRAFO_H_INCLUDED
10. #define CELLAGRAFO_H_INCLUDED
11.
12.
13. template <class tipoPeso>
14. class CellaGrafo
15. {
16. public:
17.     CellaGrafo();
18.     CellaGrafo(const CellaGrafo<tipoPeso>& );
19.     ~CellaGrafo();
20.     void setUsato(bool);
21.     void setPeso(tipoPeso);
22.     bool getUsato();
23.     tipoPeso getPeso();
24.     bool operator == ( const CellaGrafo<tipoPeso> & );
25.     bool operator != ( const CellaGrafo<tipoPeso> & );
26.     void operator = ( const CellaGrafo<tipoPeso> & );
27. private:
28.     bool usato;
29.     tipoPeso peso;
30. };
31.
32. template <class tipoPeso>
33. CellaGrafo<tipoPeso>::CellaGrafo()
34. {
35.     usato = false;
36.     peso = tipoPeso();
37. }
38.
39. template <class tipoPeso>
40. CellaGrafo<tipoPeso>::CellaGrafo(const CellaGrafo<tipoPeso>& c)
41. {
42.     usato = c.usato;
43.     peso = c.peso;
44. }
45.
46. template <class tipoPeso>
47. CellaGrafo<tipoPeso>::~CellaGrafo()
48. {
49.
50. }
51.
52. template <class tipoPeso>
53. void CellaGrafo<tipoPeso>::setUsato(bool b)

```

```

54. {
55.     usato = b;
56. }
57.
58. template <class tipoPeso>
59. void CellaGrafo<tipoPeso>::setPeso(tipoPeso p)
60. {
61.     peso = p;
62. }
63.
64. template <class tipoPeso>
65. bool CellaGrafo<tipoPeso>::getUsato()
66. {
67.     return usato;
68. }
69.
70. template <class tipoPeso>
71. tipoPeso CellaGrafo<tipoPeso>::getPeso()
72. {
73.     return peso;
74. }
75.
76. template <class tipoPeso>
77. bool CellaGrafo<tipoPeso>::operator == (const CellaGrafo<tipoPeso> & c)
78. {
79.     return peso == c.peso;
80. }
81.
82. template <class tipoPeso>
83. bool CellaGrafo<tipoPeso>::operator != (const CellaGrafo<tipoPeso> & c)
84. {
85.     return peso != c.peso;
86. }
87.
88. template <class tipoPeso>
89. void CellaGrafo<tipoPeso>::operator = (const CellaGrafo<tipoPeso> & c)
90. {
91.     peso = c.peso;
92.     usato = c.usato;
93. }
94.
95. #endif // CELLAGRAFO_H_INCLUDED

```

6.9.1.2 CellaNodo.h

```

1. /**
2.  * @file CellaNodo.h
3.  * Definizione della Cella presente nel vettore di Nodi di Grafo.h ( matrice di adiacenza)
4.  * @author Graziano Montanaro.
5.  * @date Anno Accademico 2018/19.
6. */
7.
8. #ifndef CellaNodo_H_INCLUDED
9. #define CellaNodo_H_INCLUDED
10.
11. template <class tipoElem>
12. class CellaNodo
13. {
14.     public:
15.     CellaNodo();
16.     CellaNodo(const CellaNodo<tipoElem>&);
17.     ~CellaNodo();
18.     void setPresente(bool);
19.     void setEtichetta(tipoElem );

```

```

20.     void setArchi(int);
21.     void setIndice(unsigned int);
22.     unsigned int getIndice();
23.     bool getPresente();
24.     tipoElem getEtichetta();
25.     int getArchi();
26.     bool operator == (const CellaNodo<tipoElem>&);
27.     bool operator != (const CellaNodo<tipoElem>&);
28.     void operator = (const CellaNodo<tipoElem>&);
29.
30. private:
31.     bool presente;
32.     tipoElem etichetta;
33.     int archi;
34.     unsigned int indice;
35. };
36.
37. template <class tipoElem>
38. CellaNodo<tipoElem>::CellaNodo()
39. {
40.     presente = false;
41.     etichetta = tipoElem();
42.     archi = 0;
43.     indice = 0;
44. }
45.
46. template <class tipoElem>
47. CellaNodo<tipoElem>::CellaNodo(const CellaNodo<tipoElem>& n)
48. {
49.     presente = n.presente;
50.     etichetta = n.etichetta;
51.     archi = n.archi;
52.     indice = n.indice;
53. }
54.
55. template <class tipoElem>
56. CellaNodo<tipoElem>::~CellaNodo()
57. {
58.
59. }
60.
61.
62. template <class tipoElem>
63. void CellaNodo<tipoElem>::setPresente(bool b)
64. {
65.     presente = b;
66. }
67.
68. template <class tipoElem>
69. void CellaNodo<tipoElem>::setEtichetta(tipoElem e)
70. {
71.     etichetta = e;
72. }
73.
74. template <class tipoElem>
75. void CellaNodo<tipoElem>::setArchi(int a)
76. {
77.     archi = a;
78. }
79.
80. template <class tipoElem>
81. void CellaNodo<tipoElem>::setIndice(unsigned int a)
82. {
83.     indice = a;
84. }
85.
86. template <class tipoElem>

```

```

87. unsigned int CellaNodo<tipoElem>::getIndice()
88. {
89.     return indice;
90. }
91.
92. template <class tipoElem>
93. tipoElem CellaNodo<tipoElem>::getEtichetta()
94. {
95.     return etichetta;
96. }
97.
98. template <class tipoElem>
99. bool CellaNodo<tipoElem>::getPresente()
100. {
101.     return presente;
102. }
103.
104.
105.
106. template <class tipoElem>
107. int CellaNodo<tipoElem>::getArchi()
108. {
109.     return archi;
110. }
111.
112. template <class tipoElem>
113. bool CellaNodo<tipoElem>:: operator ==(const CellaNodo<tipoElem>& n)
114. {
115.     return etichetta == n.etichetta;
116. }
117.
118. template <class tipoElem>
119. bool CellaNodo<tipoElem>:: operator !=(const CellaNodo<tipoElem>& n)
120. {
121.     return etichetta != n.etichetta;
122. }
123.
124. template <class tipoElem>
125. void CellaNodo<tipoElem>:: operator = (const CellaNodo<tipoElem>& n)
126. {
127.     presente = n.presente;
128.     etichetta = n.etichetta;
129.     archi = n.archi;
130.     indice = n.indice;
131. }
132.
133. #endif // CellaNodo_H_INCLUDED

```

6.9.2 Vettore di Liste di Adiacenza

```

1. /**
2. * @file Grafo.h
3. * Definizione struttura dati Grafo realizzata mediante Vettore di Liste di Adiacenza
4. * @author sconosciuto, modificato da Graziano Montanaro.
5. * @date Anno Accademico 2018/19.
6. */
7.
8. #ifndef _GRAFO_H
9. #define _GRAFO_H
10.
11. #include <iostream>
12. #include <cstdlib>

```

```

13. #include "Lista.h"
14. #include "Adiacenza.h"
15.
16. //definizione della classe grafo (orientato, etichettato e pesato)
17. //realizzazione tramite vettore (di lunghezza maxnodi) con liste (monodirezionali dinamiche) di adiacenza
18.
19. template<class tipoElem,class tipoPeso> class Grafo
20. {
21. public:
22.
23.     //dichiarazioni di tipo
24.     typedef unsigned int nodo; //identificatore del nodo : il nodo è identificato da un intero (indice del vettore)
25.     typedef Adiacenza<nodo, tipoPeso> adiacente; //tipo dell'elemento che farà parte della lista di adiacenza
26.     //    adiacente = (rif.nodo adiac | peso arco)
27.
28.     typedef struct //definizione dell'elemento del vettore
29.     {
30.         tipoElem etichetta; //etichetta del nodo di tipo tipoElem
31.         bool esiste; //campo booleano che indica se il nodo fa parte del grafo
32.         Lista<adiacente> adiac; //lista di adiacenza
33.     } cellaGrafo;
34.
35.
36.     Grafo(); // costruttore
37.     ~Grafo(); // distruttore
38.
39.
40.     unsigned int n_nodi();
41.
42.     //operatori di specifica
43.     void creagrafo();
44.     bool grafovusto() const;
45.     void insnodo(nodo&);
46.     void insarco(nodo, nodo);
47.     void cancnodo(nodo);
48.     void cancarco(nodo, nodo);
49.     Lista<nodo> adiacenti(nodo) const;
50.     bool esistenodo(nodo) const;
51.     bool esistearco(nodo, nodo) const;
52.     void scrivinodo(tipoElem, nodo);
53.     tipoElem legginodo(nodo) const;
54.     void scriviarco(tipoPeso, nodo, nodo);
55.     tipoPeso leggiarco(nodo, nodo);
56.
57. private:
58.     cellaGrafo table[1000]; //Dimensiono il vettore
59.     unsigned int maxnodi; //dimensione del vettore
60.     unsigned int nelementi; //indica quanti nodi effettivamente ci sono nel grafo
61.     nodo primolibero() const; //operatore ausiliare : restituisce la prima posizione libera del vettore (in modo da non avere un vettore "sparso")
62.
63.
64.
65. };
66.
67. //n viene passato nel crea grafo
68. template<class tipoElem, class tipoPeso> Grafo<tipoElem, tipoPeso>::Grafo() //costruttore specifico
69. {
70.     creagrafo();
71. }
72.
73. template<class tipoElem, class tipoPeso> Grafo<tipoElem, tipoPeso>::~Grafo() //distruttore
74. {

```

```

75.
76. }
77.
78. template<class tipoElem,class tipoPeso>
79. void Grafo<tipoElem, tipoPeso>::creagrafo() //crea il grafo
80. {
81.     maxnodi = 1000;
82.     nelementi=0; //dico che ci sono 0 nodi
83.     for (int i=0; i<1000; i++) //inizializzo il vettore mettendo a false l'appartenenza di
     tutti nodi
84.     {
85.         table[i].esiste=false;
86.     }
87. }
88.
89. template<class tipoElem,class tipoPeso>
90. bool Grafo<tipoElem, tipoPeso>::grafovusto() const //restituisce true se il grafo è vuoto,
     false altrimenti
91. {
92.     return (nelementi==0);
93. }
94.
95. template<class tipoElem,class tipoPeso>
96. void Grafo<tipoElem, tipoPeso>::insnodo(nodo &n) //inserisce il nodo che sarà identificato
     dall'indice n
97. {
98.     //passaggio del parametro per indirizzo perchè la variabile sarà modificata
99.     if (!esistenodo(n) && nelementi<1000) // precondizione nodo non appartenente
100.         {
101.             n=primolibero(); //la posizione in cui metterò il nodo del vettore sarà la
     prima libera
102.             table[n].esiste=true; //setto a true il suo campo esiste
103.             nelementi++; //aumento il contatore di nodi del grafo
104.         }
105.     }
106.
107. template<class tipoElem,class tipoPeso>
108. void Grafo<tipoElem, tipoPeso>::insarco(nodo n,nodo m) //inserisce l'arco che esce d
     al nodo n ed entra in m
109. {
110.     if (esistenodo(n) && esistenodo(m) && !esistearco(n,m))//precondizione nodi app
     artenenti e arco non esistente
111.     {
112.         typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
113.         adiacente temp; //creo l'elemento da inserire
114.         temp.scrivinodo(m); //setto l'adiacente
115.         table[n].adiac.inslista(temp,indice); //inserisco il nodo m negli adiacenti
     di n (in prima posizione)
116.     }
117. }
118.
119. template<class tipoElem,class tipoPeso>
120. void Grafo<tipoElem, tipoPeso>::cancnodo(nodo n) //elimina il nodo n
121. {
122.     if (esistenodo(n))// 1)-----
123.     {
124.         //|
125.         if (table[n].adiac.listavuota())// 2)      //|
126.         {
127.             //|
128.             bool libero=true; //|
129.             int i=0; //|
130.             while (i<maxnodi && libero) //|
131.             {
132.                 //|-
133.                 - precondizione : nodo esistente 1),che non ha archi uscenti 2) nè entranti 3)
                     if (esistenodo(i)) //|

```

```

134.          {
135.              //|
136.              libero=!esistearco(i,n); //|
137.          } //|
138.          i++; //|
139.      } //|
140.      if (libero)// 3)-----+
141.      {
142.          table[n].esiste=false; //adesso quel nodo non esiste più
143.          table[n].etichetta=NULL; //svuoto l'etichetta
144.          nelementi--; //e ho un nodo in meno nel grafo
145.      }
146.  }
147. }
148. }
149.
150. template<class tipoElem,class tipoPeso>
151. void Grafo<tipoElem, tipoPeso>::cancarco(nodo n,nodo m) //elimina l'arco che esce da n ed entra in m
152. {
153.     if (esistenodo(n) && esistenodo(m) && esistearco(n,m))//precondizione nodi appartenenti e arco esistente
154.     {
155.         bool cancellato=false;
156.         typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
157.         while (!table[n].adiac.finelista(indice) && !cancellato) //scandisco la lista finchè non cancello l'elemento
158.             {
159.                 if (table[n].adiac.leggilista(indice).legginodo()==m) //se trovo l'elemento lo cancello
160.                     {
161.                         table[n].adiac.canclista(indice); //elimino l'elemento
162.                         cancellato=true;
163.                     }
164.                 indice=table[n].adiac.succlista(indice);
165.             }
166.     }
167. }
168.
169. template<class tipoElem,class tipoPeso>
170. Lista<typename Grafo<tipoElem, tipoPeso>::nodo> Grafo<tipoElem, tipoPeso>::adiacenti(
171.     nodo n) const //restituisce la lista di adiacenti di n
172. {
173.     Lista<nodo> lista; //lista da restituire
174.     adiacente temp; //comodo
175.     if (esistenodo(n)) // precondizione nodo appartenente al grafo
176.     {
177.         typename Lista<adiacente>::posizione indice=table[n].adiac.primolista(); //indice di scansione della lista di adiacenti (lista di adiacente)
178.         typename Lista<nodo>::posizione indice2=lista.primolista(); //indice di scansione della lista di adiacendi da restituire (lista di nodo)
179.         while (!table[n].adiac.finelista(indice)) //scansione della prima lista
180.             {
181.                 temp=table[n].adiac.leggilista(indice); //lettura di adiacente
182.                 lista.inslista(temp.legginodo(),indice2); //inserisco (in coda) nella lista da restituire solo il riferimento al nodo adiacente (senza il peso dell'arco)
183.                 indice=table[n].adiac.succlista(indice);
184.                 indice2=lista.succlista(indice2);
185.             }
186.     }
187.     return(lista);
188. }
189. template<class tipoElem,class tipoPeso>
190. bool Grafo<tipoElem, tipoPeso>::esistenodo(nodo n) const //restituisce true se il nodo appartiene al grafo, false altrimenti
191. {

```

```

192.         if (n<=maxnodi) return (table[n].esiste);
193.     else return false;
194. }
195.
196. template<class tipoElem,class tipoPeso>
197. bool Grafo<tipoElem,tipoPeso>::esistearco(nodo n,nodo m) const //restituisce true se il nodo n ha un arco uscente verso m, false altrimenti
198. {
199.     bool esiste=false;
200.     if (esistenodo(n) && esistenodo(m)) //precondizione nodi appartenenti al grafo
201.     {
202.         typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
203.         while (!table[n].adiac.finelist(indice) && !esiste) //scandisco la lista finché non trovo l'elemento o è finita
204.         {
205.             if (table[n].adiac.leggilista(indice).legginodo()==m) esiste=true;
206.             indice=table[n].adiac.succlista(indice);
207.         }
208.     }
209.     return (esiste);
210. }
211.
212. template<class tipoElem,class tipoPeso>
213. void Grafo<tipoElem,tipoPeso>::scrivinodo(tipoElem val,nodo n) //scrive l'etichetta del nodo n
214. {
215.     if (esistenodo(n)) //precondizione nodo appartenente al grafo
216.         table[n].etichetta=val;
217. }
218.
219. template<class tipoElem,class tipoPeso>
220. tipoElem Grafo<tipoElem,tipoPeso>::legginode(nodo n) const //restituisce l'etichetta del nodo n
221. {
222.     tipoElem e;
223.     if (esistenodo(n)) //precondizione nodo appartenente al grafo
224.         e=table[n].etichetta;
225.     return (e);
226. }
227.
228. template<class tipoElem,class tipoPeso>
229. void Grafo<tipoElem,tipoPeso>::scriviarco(tipoPeso val,nodo n,nodo m) //scrive il peso dell'arco che va dal nodo n al nodo m
230. {
231.     if (esistenodo(n) && esistenodo(m)) //precondizione nodi appartenenti al grafo (c'è anche arco appartenente ma non conviene altrimenti c'è una scansione solo per vedere)
232.     {
233.         //se esiste e poi si farebbe la seconda scansione per aggiornare
234.         bool aggiornato=false;
235.         adiacente temp(m,val);
236.         typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
237.         while (!table[n].adiac.finelist(indice) && !aggiornato) //scandisco la lista finché non trovo l'elemento o è finita
238.         {
239.             if (table[n].adiac.leggilista(indice).legginodo()==m)
240.             {
241.                 table[n].adiac.scrivilista(temp,indice);
242.                 aggiornato=true;
243.             }
244.             indice=table[n].adiac.succlista(indice);
245.         }
246.     }
247. }
248.

```

```

249.     template<class tipoElem,class tipoPeso>
250.         tipoPeso Grafo<tipoElem,tipoPeso>::leggiarco(nodo n,nodo m) //restituisce il peso d
ell'arco che va da n a m
251.     {
252.         tipoPeso a;
253.         if (esistenodo(n) && esistenodo(m)) //precondizione nodi appartenenti al grafo
(c'è anche arco appartenente ma non conviene altrimenti c'è una scansione solo per veder
e
254.         {
255.             //se esiste e poi si farebbe la seconda scansione per la lettura
//quindi si fa un'unica scansione in
cui si ricerca e si legge)
256.             bool letto=false;
257.             typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
258.             while (!table[n].adiac.finelista(indice) && !letto) //scandisco la lista fi
nchè non trovo l'elemento o è finita
259.             {
260.                 if (table[n].adiac.leggilista(indice).legginodo()==m)
261.                 {
262.                     a=table[n].adiac.leggilista(indice).leggipeso();
263.                     letto=true;
264.                 }
265.                 indice=table[n].adiac.succlista(indice);
266.             }
267.         }
268.         return(a);
269.     }
270.
271.
272.     //operatori ausiliari
273.
274.     template<class tipoElem,class tipoPeso>
275.         unsigned int Grafo<tipoElem,tipoPeso>::primolibero() const //restituisce la prima p
osizione del vettore libera
276.     {
277.         nodo i=-1;
278.         bool libero=false;
279.         while (!libero) //scorre il vettore finchè non trova una posizione libera
280.         {
281.             //il controllo è solo su libero perchè so che mi fermerò per forza poichè q
uesto metodo viene chiamato solo se il vettore non è pieno
282.             i++;
283.             libero=!table[i].esiste;
284.         }
285.         return (i);
286.     }
287.
288.
289.     template<class tipoElem,class tipoPeso> unsigned int Grafo<tipoElem,tipoPeso>::n_no
di()
290.     {
291.         return nelementi;
292.     }
293.
294.
295. #endif

```

6.9.2.1 Adiacenza.h

```

1. #ifndef _ADIACENZA_H
2. #define _ADIACENZA_H
3.
4. #include <iostream>
5. #include <cstdlib>
6.

```

```

7. using namespace std;
8.
9. //tipo dell'elemento che sarà usato nella lista di adiacenza del grafo pesato
10.
11. template<class nodo,class tipoPeso> class Adiacenza
12. {
13. public:
14.
15.
16.     //costruttori (generico di default)
17.     Adiacenza();
18.     Adiacenza(nodo,tipoPeso);
19.     ~Adiacenza();
20.     //distruttore di default
21.
22.     //setter e getter
23.     void scrivinodo(nodo);
24.     nodo legginodo() const;
25.     void scrivipeso(tipoPeso);
26.     tipoPeso leggipeso() const;
27.
28.
29. private:
30.     nodo adiacente; //riferimento al nodo adiacente
31.     tipoPeso peso; //peso dell'arco
32.
33. };
34.
35.
36. template <class nodo,class tipoPeso> Adiacenza<nodo,tipoPeso>::Adiacenza() //costruttore generico
37. {
38. }
39.
40. template <class nodo,class tipoPeso> Adiacenza<nodo,tipoPeso>::Adiacenza(nodo n, tipoPeso p) //costruttore specifico
41. {
42.     adiacente=n;
43.     peso=p;
44. }
45.
46. template <class nodo,class tipoPeso> Adiacenza<nodo,tipoPeso>::~Adiacenza()
47. {
48.     //dtor
49. }
50.
51. template <class nodo,class tipoPeso> void Adiacenza<nodo,tipoPeso>::scrivinodo(nodo n)
52. {
53.     adiacente=n;
54. }
55.
56. template <class nodo,class tipoPeso> void Adiacenza<nodo,tipoPeso>::scrivipeso(tipoPeso p)
57. {
58.     peso=p;
59. }
60.
61. template <class nodo,class tipoPeso> nodo Adiacenza<nodo,tipoPeso>::legginodo() const
62. {
63.     return(adiacente);
64. }
65.
66. template <class nodo,class tipoPeso> tipoPeso Adiacenza<nodo,tipoPeso>::leggipeso() const
67. {
68.     return(peso);

```

```
69. }
70.
71.
72. //sovraffigura output
73.
74. template<class nodo, class tipoPeso> ostream& operator<<(ostream& os, const Adiacenza<nodo,
    tipoPeso>& a)
75. {
76.     os<<"(" <<a.legginodo()<<" | "<<a.leggipeso()<<")";
77.     return(os);
78. }
79.
80. #endif
```