



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	16
1.1	Modifica Pastorelli Antonio	16
1.2	Modifica Basile Antonio.....	16
1.3	Modifica Del Giudice Angelo.....	16
1.4	Modifica Moschetti Marco	16
1.5	Modifica Germano Galeandro.....	16
1.6	Modifica D'Andria Ivan – Dresden Claudio	17
1.7	Modifica Federica Forte.....	17
1.8	Modifica Gianluca Vacca.....	17
1.9	Modifica Salvatore Vestita	17
1.10	Modifica Vincenzo Giannuzzo	17
1.11	Modifica Cicala Giacomo	17
1.12	Modifica Scatigna Gianluca.....	17
1.13	Modifica Della Folgore Grazia	17
1.14	Modifica Palagiano Marcello.....	18
1.15	Modifiche Chiarappa Rosa	18
1.16	Modifiche Disabato Margherita	18
1.17	Modifiche Zagaria Marta	18
1.18	Modifiche Castellana Giovanni	18
1.19	Modifiche Collocola Flavio	18
1.20	Modifiche Lentini Giovanni.....	18
1.21	Modifiche Spina Alex	18
1.22	modifiche pierluigi bemporato	19
1.23	MODIFICHE SIMONE BAGORDO.....	19
1.24	MODIFICHE Samuele Pesce.....	19
1.25	MODIFICHE Davide Rano	19
1.26	modifiche leandra palemburgi.....	19
1.27	MODIFICHE SIMONE VERSIENTI	19
1.28	MODIFICHE DE FLORIO CRISTINA.....	20
1.29	MODIFICHE COSTANTINI ANDREA.....	20
1.30	MODIFICHE BASO MATTEO.....	20
1.31	MODIFICHE GRAVINA ANTONIO	20
2	Analisi	20
2.1	Museo	21
2.2	Scuola e Aule	21
2.3	Archivio e Museo (vecchia versione del museo, inserita per completezza)	21

2.4	Aula singola (incompleta)	22
2.5	Ufficio Postale (incompleta).....	22
2.6	Dialoghi.....	22
2.7	Sala scommesse (incompleta).....	22
2.8	Simulatore (incompleta)	22
2.9	Auditorium.....	23
2.10	Stazione (incompleta).....	23
2.11	Biblioteca (incompleta).....	23
2.12	Bari – Roma – Pisa (incompleta).....	23
2.13	Meccanico (incompleta)	24
2.14	Verbo "ruba", personaggio "carabiniere",luogo "caserma" e oggetto cassaforte	24
2.15	Ristorante e Fast Food	24
2.16	Estrazione di un personaggio casuale.....	25
2.17	Chiesa	26
2.18	Ufficio / Deposito	29
2.19	Sali&Tabacchi	29
2.20	Distanza/Sforzo	29
2.21	Fabbrica	30
2.22	TELETRASPORTA	30
2.23	Cassaforte e cassetta di sicurezza malmessa	30
2.24	Verbo "ritorna" e verbo "resoconto"	30
2.25	Laboratorio analisi.....	30
2.26	GUARDA MAPPA	31
2.27	CUCINA	31
2.28	CAPACITÀ	31
3	Progettazione	31
3.1	Museo	31
3.1.1	Struttura dati	32
3.2	Scuola e Aule	32
3.2.1	Luogo Scuola.....	32
3.2.2	Luogo Classe 1A.....	32
3.2.3	Luogo Classe 2A	32
3.2.4	Luogo Classe 3A	33
3.2.5	Struttura dati	33
3.3	Archivio e Museo (vecchia versione del museo)	33
3.3.1	Luogo Archivio	33

3.3.2	File.....	33
3.3.3	Museo	33
3.3.4	Struttura dati	34
3.4	Aula singola	34
3.4.1	Computer	34
3.4.2	Sedie.....	34
3.4.3	Proiettore	35
3.4.4	Libri	35
3.4.5	Lezione	35
3.4.6	Struttura dati	35
3.5	Ufficio Postale	35
3.5.1	Struttura dati	36
3.6	Dialoghi.....	36
3.6.1	Struttura dati	37
3.7	Sala scommesse e simulatore	37
3.7.1	Azioni	37
3.7.2	Inserimento Azione "Guarda simulatore"	38
3.7.3	Inserimento Azione "Avvia simulatore" e "Gioca simulatore"	38
3.7.4	Struttura dati	38
3.8	Auditorium.....	38
3.8.1	Struttura dati	39
3.9	Stazione	39
3.9.1	Struttura dati	40
3.10	Palestra	40
3.10.1	Struttura dati	41
3.11	Sala giochi	41
3.11.1	Slot machine.....	42
3.11.2	Struttura dati	42
3.12	Biblioteca	43
3.12.1	Struttura dati	43
3.13	Bari – Roma – Pisa	43
3.13.1	Struttura dati	43
3.14	Meccanico	43
3.14.1	Struttura dati	44
3.15	creazione verbo “ruba”, personaggio “carabiniere” luogo “caserma” e oggetto cassaforte	45
3.15.1	Struttura dati	45

3.16	Ristorante e Fast Food	46
3.16.1	Struttura dati	46
3.17	Estrazione di un personaggio casuale.....	47
3.18	Chiesa	48
3.18.1	Struttura dati	56
3.19	Ufficio / deposito.....	57
3.19.1	Documenti.....	57
3.19.2	Struttura dati	57
3.20	Sali&Tabacchi	57
3.21	Distanza/Sforzo	58
3.21.1	Struttura dati	59
3.22	Fabbrica	59
3.22.1	Struttura dati	60
3.23	TELETRASPORTA	60
3.23.1	Struttura dati	61
3.24	Cassaforte e cassetta di sicurezza malmessa	61
3.24.1	Struttura dati	61
3.25.1	Struttura dati	61
3.26	Laboratorio analisi.....	62
3.26.1	Struttura dati	62
3.27	guarda mappa	62
3.27.1	Struttura dati	62
3.28	CAPACITÀ	62
3.28.1	Struttura dati	63
4	Realizzazione.....	63
4.1	Museo	63
4.2	Scuola e Aule	65
4.2.1	Scuola e le classi 1A, 2A, 3A	65
4.2.2	Oggetti	65
4.2.3	Azioni	66
4.3	Archivio e Museo (vecchia versione del museo).....	66
4.3.1	Archivio e Museo	67
4.3.2	Oggetti	67
4.3.3	Azioni	68
4.4	Aula singola	68
4.4.1	Aula	68

4.4.2	Oggetti	69
4.4.3	Azioni	69
4.4.4	Comandi	70
4.5	Ufficio Postale	70
4.5.1	Mappa.nav	70
4.5.2	24.txt	71
4.5.3	Inserimento oggetto "Sportello Telematico"	71
4.5.4	Inserimento oggetto "Documento d'Identità"	72
4.5.5	Inserimento oggetto "Lettera"	72
4.5.6	Inserimento oggetto "Pacco"	73
4.6	Dialoghi	73
4.7	Sala scommesse e simulatore	74
4.7.1	Aggiunta oggetti e vocaboli nel gioco	74
4.8	Auditorium	74
4.8.1	Classi create	74
4.8.2	Realizzazioni usate	75
4.8.3	Modifica Mappa	75
4.8.4	Oggetti	76
4.8.5	Azioni	76
4.9	Stazione	77
4.10	Palestra	78
4.10.1	Oggetti	79
4.10.2	Azioni	79
4.10.3	Comandi	80
4.11	Sala Giochi	80
4.11.1	Oggetto	80
4.11.2	Azioni	81
4.11.3	Comandi	81
4.12	Biblioteca	82
4.12.1	Oggetti	82
4.12.2	Azioni	82
4.12.3	Comandi	82
4.13	Bari – Roma – Pisa	83
4.13.1	Oggetti	84
4.13.2	Azioni	84
4.14	Modifica codici luoghi – Palagiano Marcello	84

4.14.1	Modifica Mappa.....	85
4.14.2	Modifica MappaOsservatorio.....	85
4.14.3	Modifica MappaAliena.....	85
4.15	Meccanico	85
4.15.1	Oggetti	86
4.15.2	Azioni	86
4.16	CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE”, LUOGO “CASERMA” E OGGETTO CASSAFORTE	87
4.16.1	Verbi “ruba” e “rubati” e azione “apri cassaforte”	87
4.16.2	Personaggio “carabiniere”	88
4.16.3	Luogo “caserma”	89
4.17	Chiesa	89
4.17.1	Aggiunta oggetti e vocaboli nel gioco	89
4.18	Ufficio / deposito.....	90
4.18.1	Oggetti	90
4.18.2	Azioni	90
4.18.3	Comandi	90
4.19	Sali & Tabacchi	91
4.19.1	Oggetti	91
4.19.2	Azioni	92
4.19.3	Comandi	92
4.20	Distanza/Sforzo	93
4.21	Fabbrica	94
4.21.1	Inserimento luogo nella mappa.....	94
4.21.2	Oggetti	95
4.21.3	Azioni	95
4.22	TELETRASPORTA.....	95
4.22.1	Verbo “teletrasporta”	95
4.22.2	Azione teletrasporta	96
4.23	Cassaforte e cassetta di sicurezza malmessa	97
4.23.1	Luogo “Cassaforte”	97
4.23.2	Oggetto “Cassetta di sicurezza malmessa”	98
4.23.3	Azioni “Deposita oggetto”, “Controlla cassetta”, “Ritira oggetto”	98
4.24	Errore. L'origine riferimento non è stata trovata.....	98
4.24.1	Azione ritorna	99
4.24.2	Azione resoconto	99
4.25	Laboratorio analisi.....	100

4.25.1	Luogo.....	100
4.25.2	Oggetti	100
4.25.3	Azioni	100
4.26	GUARDA MAPPA	101
4.26.1	Verbo “guarda mappa”	101
4.26.2	Azione “guarda mappa”	101
4.27	CUCINA	102
4.27.1	Oggetti	102
4.27.2	AZIONI.....	103
4.28	CAPACITÀ	103
4.28.1	MAPPATURA.....	104
4.28.2	VOCABOLI.....	104
4.28.3	OGGETTI	104
4.28.4	COMANDI.....	104
5	Implementazione	105
5.1	Museo.....	105
5.1.1	Museo.h	105
5.1.2	Museo.cpp.....	106
5.1.3	Lista.h.....	120
5.1.4	Cella_L_DP.h.....	125
5.1.5	Grafo.h	128
5.1.6	Adiacenza.h	137
5.1.7	Modifiche relative ad Astro.cpp	140
5.2	Scuola e Aule	144
5.2.1	Astro.h.....	144
5.2.2	Gioco.h	145
5.2.3	Astro.cpp	145
5.2.4	Mappa.nav	163
5.2.5	Aggiunta dei file nella cartella descrizioni.....	164
5.3	Archivio e Museo (vecchia versione del museo).....	165
5.3.1	Astro.h.....	165
5.3.2	Gioco.h	166
5.3.3	Astro.cpp	166
5.3.4	Gioco.cpp	181
5.3.5	Mappa.nav	208
5.3.6	Descrizioni	209

5.4	Aula singola	210
5.4.1	AGGIORNAMENTO DEI VOCABOLI DEL DIZIONARIO.....	210
5.4.2	Aggiornamento del dizionario degli oggetti.....	210
5.4.3	Aggiornamento delle azioni di gioco.....	211
5.4.4	Implementazione dell'Aula	221
5.4.5	Correzione Funzionalità usa computer (Andrea Tursi)	222
5.5	Ufficio Postale	223
5.5.1	Inserimento del luogo "Ufficio Postale"	223
5.5.2	Gioco.h	226
5.5.3	Codice per inserimento oggetti	226
5.5.4	Nuove azioni	226
5.5.5	Modifiche al progetto di Bellanova	235
5.6	Dialoghi	236
5.6.1	Implementazione delle variabili	236
5.6.2	Implementazione dei metodi	236
5.6.3	Modifiche dei metodi già presenti in Gioco.cpp	240
5.6.4	File aggiunti	250
5.7	Sala scommesse e simulatore	264
5.7.1	Implementazione luogo Sala scommesse e nuove azioni nel gioco	264
5.7.2	Strutture dati aggiuntive	269
5.8	Auditorium	275
5.8.1	Gioco.h	275
5.8.2	Astro.h.....	275
5.8.3	Astro.cpp	276
5.8.4	Gioco.cpp	284
5.9	Stazione	285
5.9.1	Mappa.nav	285
5.9.2	Descrizioni (33.txt).....	286
5.9.3	Astro.cpp	286
5.9.4	Astro.h.....	293
5.9.5	Biglietto.h.....	293
5.9.6	Biglietto.cpp	294
5.10	Palestra	295
5.10.1	Mappa.nav	295
5.10.2	Descrizioni (34.txt).....	295
5.10.3	Astro.cpp	296

5.10.4	Astro.h.....	298
5.10.5	Gioco.cpp	298
5.10.6	Gioco.h	299
5.10.7	StatoFisico.h	300
5.10.8	StatoFisico.cpp.....	301
5.10.9	Palestra.h	302
5.10.10	Scheda.h.....	303
5.11	Sala giochi	303
5.11.1	Aggiornamento mappa	303
5.11.2	Aggiornamento dei vocaboli	304
5.11.3	Aggiornamento azioni.....	304
5.11.4	Aggiornamento oggetti.....	305
5.11.5	Aziona gioca slot-machine.....	305
5.11.6	Metodo load() e save().....	311
5.12	Implementazione Biblioteca	312
5.12.1	Aggiornamento mappa	312
5.12.2	Aggiornamento dei vocaboli	312
5.12.3	Aggiornamento azioni.....	312
5.12.4	Aggiornamento oggetti	312
5.12.5	Dichiarazione azioni	313
5.12.6	Implementazione metodi per funzionamento biblioteca	316
5.12.7	ChangeLog	322
5.13	Implementazione Bari – Roma – Pisa	327
5.13.1	Mappa.nav	328
5.13.2	Descrizioni(cartella)	328
5.13.3	Astro.cpp	329
5.13.4	Astro.h.....	337
5.13.5	Gioco.cpp	337
5.13.6	Gioco.h	338
5.13.7	Luogo.cpp.....	338
5.13.8	Luogo.h.....	339
5.13.9	Oggetti.cpp.....	339
5.13.10	Oggetti.h.....	339
5.13.11	Oggetto.h.....	340
5.13.12	Oggetto.cpp.....	340
5.13.13	Veicolo.h.....	340

5.13.14	Veicolo.cpp	341
5.13.15	Bagagliaio.h	344
5.13.16	Bagagliaio.cpp.....	345
5.13.17	Autobus.h	348
5.13.18	Autobus.cpp.....	349
5.13.19	Automobile.h	350
5.13.20	Automobile.cpp	351
5.14	Modifica codici luoghi – Palagiano Marcello	352
5.14.1	Modifica file “Mappa.nav”	352
5.14.2	Modifica file “MappaOsservatorio.nav”	353
5.14.3	Modifica file “MappaAliena.nav”	354
5.14.4	Modifica file “Trasporti.nav”	355
5.14.5	Modifica righe di codice	355
5.15	Implementazione Meccanico	360
5.15.1	Aggiunta del file “Batteria.h”	361
5.15.2	Aggiunta del file “Batteria.cpp”	361
5.15.3	Aggiunta del file “Attivita.h”	362
5.15.4	Aggiunta del file “Attivita.cpp”	363
5.15.5	Modifica al file “Gioco.cpp”	364
5.15.6	Modifica al file “Gioco.h”	365
5.15.7	Modifica al file “Astro.h”	366
5.15.8	Modifica al file “Astro.cpp”	367
5.15.9	Modifica al file “Mappa.nav”	379
5.16	CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE”, LUOGO “CASERMA” E OGGETTO CASSAFORTE	380
5.16.1	Verbo “ruba”	380
5.16.2	Verbo “rubati”	380
5.16.3	Personaggio “carabiniere”	380
5.16.4	Luogo “caserma”	388
5.17	Ristorante e Fast Food	391
5.17.1	Implementazione variabile sazietà	391
5.17.2	Aggiornamento del dizionario	392
5.17.2.1.	Parole per il controllo della sazietà	392
5.17.2.2.	Parole per l’acquisto delle pietanze	392
5.17.2.3.	Aree Ristoro	392
5.17.2.4.	Parole per controllare lo zaino e la tasca	392
5.17.2.5.	Parole inerenti al buono pasto	392

5.17.2.6.	Parole inerenti agli alimenti delle zone ristoro	392
5.17.3	Aggiornamento del dizionario degli oggetti.....	393
5.17.3.1.	Oggetti "zaino termico" e "tasca"	393
5.17.3.2.	Inserimento degli alimenti nel dizionario degli oggetti	393
5.17.3.3.	Inserimento dei buoni pasto nel dizionario degli oggetti	394
5.17.4	Aggiunta delle azioni di gioco.....	394
5.17.4.1.	Azione per il controllo della sazietà	394
5.17.4.2.	Azioni per l'acquisto di cibo nel fast food e nel ristorante.....	394
5.17.4.3.	Azioni per la consumazione del cibo presenti nello zaino termico.....	394
5.17.4.4.	Azioni per la consumazione della tasca e dello zaino	395
5.17.5	Classe Oggetto: Modifiche da effettuare	395
5.17.5.1.	Costruttore per i buoni pasto	395
5.17.5.2.	Costruttore per i cibi	396
5.17.5.3.	Implementazione del metodo get_categoria.....	396
5.17.5.4.	Implementazione del metodo get_valore_e	397
5.17.6	File Mappa.nav: Modifiche effettuate	397
5.17.7	Procedure.....	398
5.17.7.1.	Monitoraggio della sazietà	398
5.17.7.2.	Gestione del tempo rimanente	399
5.17.7.3.	Gestione degli acquisti del cibo	400
5.17.7.4.	Miglioramento del livello energetico	401
5.17.7.5.	Visualizzazione del contenuto della tasca	402
5.17.7.6.	Visualizzazione del contenuto dello zaino.....	403
5.17.8	Altre modifiche	404
5.17.8.1.	Modifica del metodo lascia	404
5.17.8.2.	Modifiche al metodo init.....	405
5.17.8.3.	Modifiche al metodo prendi.....	405
5.17.8.4.	Dichiarazione delle nuove procedure	407
5.17.8.5.	Modifica del metodo elenca_oggetti	408
5.18	Estrazione di un personaggio casuale.....	409
5.19	Chiesa	433
5.19.1	Implementazione modifiche Stato Salute	433
5.19.2	Implementazione luogo "Chiesa" e nuove azioni di gioco	434
5.20	Ufficio / deposito.....	439
5.20.1	Implementazione consenga documento.....	439
5.20.2	Implementazione ritira documento.....	440

5.21	Sali&Tabacchi	440
5.21.1	Mappa.nav	441
5.21.2	43.txt	441
5.21.3	Astro.h.....	441
5.21.4	Astro.cpp.....	442
5.21.5	Gioco.h.....	457
5.21.6	Gioco.cpp	457
5.21.7	Interfaccia.h.....	458
5.21.8	Interfaccia.cpp	458
5.21.9	Scontrino.h	458
5.21.10	Scontrino.cpp.....	459
5.21.11	Risoluzione Bug Del Portafoglio.....	459
5.22	Distanza/Sforzo	460
5.22.1	Contapassi.cpp.....	460
5.22.2	Mappa.h.....	461
5.22.3	Mappa.cpp	461
5.22.4	Gioco.h	463
5.22.5	Gioco.cpp	463
5.22.6	Astro.h.....	463
5.22.7	Astro.cpp	464
5.22.8	Grafo.h	464
5.22.9	Nodo_Grafo.h.....	469
5.22.10	Nodo_Grafo.cpp	470
5.23	Fabbrica	471
5.23.1	Mappa.nav	471
5.23.2	Astro.cpp	471
5.23.3	Fabbrica.h.....	473
5.23.4	Fabbrica.cpp	474
5.24	Funzione Teletrasporta.....	478
5.25	Cassaforte e cassetta di sicurezza malmessa, correzione verbo “usa”, “Parla”, indici mappe e oggetti, tessera	480
5.26	Verbo “ritorna” e verbo “resoconto”	485
5.28	Laboratorio analisi.....	492
5.28.1	Interfaccia.cpp (123-126)	492
5.28.2	Mappa.nav (1, 21, 50, 133, 134)	493
5.28.3	Descrizioni	493
5.28.4	Astro.h (310).....	493

5.28.5	Astro.cpp	493
5.28.6	Dizionario.h	496
5.29	Funzione GUARDA MAPPA	500
5.30	Luogo CUCINA	502
	modifica Astro.h	503
	modifica Astro.cpp	503
	Modifica Vocabolario	503
	modifica oggetti	503
	modifica azioni	503
	modifica switch case	503
	modifica Mappa.nav	504
	Cucina.h	505
	Cucina.cpp	505
	Implementazione di AggiungiCibo	505
	Implementazione di usaFrigo	506
	Implementazione di guardaTelevisione	506
	Implementazione di preparaCibo	507
5.31	Comando “oggetti”	507
5.32	Funzionalità “zaino termico”	508
5.33	Implementazione delle capacità	509
	Implementazione di Capacita.h:	509
	Implementazione di Capacita.cpp:	510
	Implementazione di Capacitasingola.h:	513
	Implementazione di Capacitasingola.cpp:	514
	Azioni aggiunte in Astro.cpp:	515
	Modifiche in Astro.h:	524
	Modifiche in Gioco.h:	524
	Modifiche in Gioco.cpp:	524
6	Strutture intercambiabili	529
6.1	<i>Lista</i>	529
6.1.1	Lista con Puntatore	529
6.1.2	Lista con Vettore sequenziale	535
6.1.3	Lista con puntatori doppi	538
6.2	Lista Ordinata	542
6.2.1	Lista Ordinata con vettore:	542
6.2.2	Lista Ordinata con puntatori:	544

6.3	Pila	545
6.3.1	Pila con puntatori:	545
6.3.2	Pila con cursori	547
6.3.3	Pila con lista.....	549
6.4	Coda.....	551
6.4.1	Coda con lista	552
6.4.2	Coda con puntatori.....	553
6.4.3	Coda con Vettore sequenziale.....	555
6.5	<i>Coda di priorità</i>	557
6.5.1	Coda con priorità con vettore (Heap)	557
6.5.2	Coda con priorità con lista ordinata	562
6.6	Albero n-ario	565
6.6.1	AlberoNario: realizzazione con alberi binari	565
6.6.2	AlberoNario: realizzazione primofiglio/fratello.....	575
6.6.3	ALBERO N-ARIO REALIZZATO CON VETTORE DI PADRI	578
6.7	<i>Insiemi</i>	582
6.7.1	Insiemi con lista	582
6.8	Dizionario	586
6.8.1	Dizionario con vettore ordinato	586
6.8.2	Dizionario con vettore non ordinato	589
6.8.3	Dizionario con lista.....	591
6.8.4	Dizionario con array con Lista di trabocco	593
6.9	Grafi	595
6.9.1	Matrice di Adiacenza	595
6.9.2	Vettore di Liste di Adiacenza	613
6.10	Alberi Binari.....	620
6.10.1	REALIZZAZIONE CON CURSORI DELL'ALBERO BINARIO	620
6.10.2	REALIZZAZIONE CON VETTORE	623
7	Tabella di Test delle funzionalità	628
8	Porting da Windows a Linux	631
8.1	Codifica	631
1)	Testing	632
2)	Aprire correttamente il progetto	634

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 PASTORELLI ANTONIO

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 BASILE ANTONIO

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'avventura 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 FOLGORE 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 Folgore 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 DISABATO MARGHERITA

Si richiede di integrare nel progetto di Zagaria Marta le parti presenti nel progetto di Gallone Gianmarco che mancano, uniformando le strutture dati presenti ed eventualmente aggiungendo nuove realizzazioni.

1.17 MODIFICHE ZAGARIA MARTA

Si richiede di aggiungere al progetto “Adventure” di Castellana Giovanni la funzionalità del progetto di Castellana Valerio, che prevede l'estrazione di un personaggio casuale da un elenco di personaggi nei luoghi dell'avventura.

Il personaggio presente nella stanza avrà l'obiettivo di favorire o penalizzare il giocatore nel corso della partita tramite frasi casuali che potranno allontanare l'avventuriero dalla soluzione o aiutarlo.

Nota: Il personaggio sarà visibile solo per quella mossa.

1.18 MODIFICHE CASTELLANA GIOVANNI

Si richiede di modificare il progetto base di Michele Albano con l'aggiunta di un nuovo luogo identificato come "Chiesa" in cui il personaggio dovrà fare qualcosa per poter risolvere il gioco.

1.19 MODIFICHE COLLOCOLA FLAVIO

Si richiede di risistemare tramite il progetto di Olimpio Luis e Farina Alessio le funzionalità “ristorante” ed “ufficio” non funzionanti nel progetto base di Balzano Nicola e Miccoli Vincenzo.

1.20 MODIFICHE LENTINI GIOVANNI

Si richiede di integrare nel progetto di Balzano e Miccoli il luogo aggiunto da Joseph Tortorella “sali&tabacchi”. Oltre ad integrare nel progetto base le funzionalita' di Joseph Tortorella, è richiesto inoltre di integrare le realizzazioni presenti nella cartella "Strutture Dati Intercambiabili" di Joseph Tortorella in quelle del progetto base.

1.21 MODIFICHE SPINA ALEX

Si richiede di integrare la funzionalità di calcolo della distanza e dello sforzo, presente nel progetto di Andrea De Rinaldis, all'interno del progetto base di Giovanni Lentini.

Successivamente all'integrazione, dovrà essere garantito il corretto funzionamento della nuova operazione e, ovviamente, delle altre precedentemente presenti all'interno del progetto base.

Inoltre, è richiesta un'analisi e una valutazione di alcune strutture dati intercambiabili presenti nel progetto di De Rinaldis.

1.22 MODIFICHE PIERLUIGI BEMPORATO

È stato richiesto di aggiungere un luogo chiamato "Fabbrica" in cui il protagonista possa costruire degli oggetti, aggiungendo i pezzi necessari in una determinata sequenza. Inoltre, una volta che la costruzione è avvenuta con successo, il protagonista guadagna denaro in base a cosa è stato costruito.

1.23 MODIFICHE SIMONE BAGORDO

Si richiede di integrare le funzionalità realizzate da Giuseppe Agnusdei nel progetto base di Bemporato. L'aggiunta di una nuova funzionalità chiamata "teletrasporto", con cui il personaggio potrà muoversi tra i vari luoghi, in maniera istantanea, proseguire verso nuovi luoghi e ritornare in ordine inverso nei luoghi in cui si è utilizzato precedentemente il comando. È richiesto inoltre di integrare le realizzazioni presenti nella cartella "Strutture Dati Intercambiabili" di Giuseppe Agnusdei in quelle del progetto base.

1.24 MODIFICHE SAMUELE PESCE

È stato richiesto di integrare le funzionalità realizzate da Roberto Carlucci nel progetto base di Simone Bagordo: l'aggiunta di un nuovo luogo "Cassaforte" contenente una cassetta di sicurezza malmessa, che non richiede chiavi o codici per essere utilizzata, nella quale il giocatore può depositare e ritirare degli oggetti. È richiesto inoltre di integrare le realizzazioni presenti nella cartella "Strutture Dati Intercambiabili" di Roberto Carlucci in quelle del progetto base.

1.25 MODIFICHE DAVIDE RANO

E' stato richiesto di integrare il luogo caserma del progetto del collega Senapo nel progetto base, unificando la caserma già presente nel progetto base con quella del collega Senapo. Aggiungendo una cassaforte in grado di poter recuperare gli oggetti che vengono confiscati dal carabiniere una volta arrestati. Ed utilizzare una qualche struttura dati per estendere la caserma del collega Senapo.

1.26 MODIFICHE LEANDRA PALEMBURGI

1.27 MODIFICHE SIMONE VERSIENTI

E' stato richiesto di integrare il luogo "Laboratorio di analisi", originariamente inserito del collega Scarci ed ottimizzato dalla collega Bottiglione. In più, si richiede di integrare le realizzazioni presenti nella cartella "Strutture dati intercambiabili" della collega Bottiglione nel progetto base.

1.28 MODIFICHE DE FLORIO CRISTINA

Si richiede di integrare la funzionalità realizzata da Fiore Zaccaria nel progetto base. L'aggiunta di una nuova funzionalità chiamata “guarda mappa”, con cui il giocatore può visualizzare o una porzione di mappa adiacente al luogo in cui si trova o può scegliere di visualizzare l'intera mappa di gioco.

E' inoltre richiesto di integrare le realizzazioni presenti nella cartella “Strutture dati intercambiabili” di Fiore Zaccaria nel progetto base.

1.29 MODIFICHE COSTANTINI ANDREA

Si richiede di integrare la funzionalità di Gaudio Giorgio nel progetto base che prevede l'aggiunta del luogo “Cucina” con le relative funzionalità. Si richiede, inoltre, di integrare le strutture dati intercambiabili realizzate dal collega Gaudio non presenti nel progetto base.

1.30 MODIFICHE BASO MATTEO

Si richiede di integrare le funzionalità realizzate da Perrone Emanuela nel progetto base che prevedono l'aggiunta di “zaino termico” e del comando “oggetti”. Si richiede, inoltre, di integrare le strutture dati intercambiabili realizzate dal collega Perrone non presenti nel progetto base.

1.31 MODIFICHE GRAVINA ANTONIO

La traccia richiede l'integrazione delle funzionalità integrate dal collega Vergine Erik nel progetto base2012 del collega Bagordo. In particolare, si dovranno aggiungere le capacità per il personaggio. Si richiede anche l'integrazione delle strutture dati presenti nei due progetti nelle sottocartelle opportune di “Strutture Dati Intercambiabili”, per poi verificarne l'intercambiabilità e modificarle in caso di errori. Si possono anche proporre nuove realizzazioni.

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 (INCOMPLETA)

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 (INCOMPLETA)

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 (INCOMPLETA)

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 (INCOMPLETA)

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 (INCOMPLETA)

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 (INCOMPLETA)

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 (INCOMPLETA)

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 (INCOMPLETA)

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 “CARABINIÈRE”, LUOGO “CASERMA” E OGGETTO CASSAFORTE

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.

Nella caserma il giocatore troverà una cassaforte in cui potrà rubare di nuovo gli oggetti che sono stati sequestrati dal carabiniere dopo l'arresto. La cassaforte potrà essere aperta con un codice presente nel gioco. Nella caserma verranno inserite delle armi (inutilizzabili) visibili dal giocatore. Ed una cella per arricchire il luogo.

2.15 RISTORANTE E FAST FOOD

In base alla modifica richiesta, si aggiungeranno due nuovi luoghi, Fast Food e Ristorante, in cui il giocatore potrà placare l'appetito del comandante, acquistando del cibo attraverso dei buoni pasto, poiché uno degli aspetti principali della modifica sarà quello di introdurre la gestione dell'appetito del protagonista di questa avventura. Fin dall'inizio, il personaggio avvertirà un senso di fame costantemente crescente, che potrà essere sempre controllato dal giocatore visualizzando il livello di sazietà.

2.16 ESTRAZIONE DI UN PERSONAGGIO CASUALE

Nel mio caso, partendo da un' avventura già scritta mi è stato chiesto di aggiungere la classe personaggi:

Il personaggio sarà presente nell' avventura sin da subito e sarà possibile interagire con egli, attraverso degli opportuni comandi.

Ci saranno molti personaggi con i quali si potrà interagire, quando si entrerà in un luogo ci sarà un personaggio, ma quest' ultimo sarà presente nel medesimo luogo solo per quella mossa, dopodichè verrà eliminato dal gioco; per questo motivo è stato utile aggiungere 50 mosse per non complicare troppo l' avventura.

Il personaggio interagirà con il protagonista comunicandogli delle frasi, che potranno essere di aiuto, o potranno portarlo ad avere complicazioni nel gioco.

Ogni personaggio avrà a disposizione 2 frasi (una di intralcio e una di aiuto), la comparsa di una delle due sarà dettata dal caso.

Inoltre, alcuni personaggi avranno a disposizione o due o quattro frasi in più, che in base agli oggetti posseduti, cambieranno la frase sottoposta al giocatore.

I personaggi potranno comparire nei seguenti luoghi: la cabina del pilota, la cabine del secondo pilota, nella cabina di pilotaggio, nel corridoio e alle sue estremità, nel compartimento stagno, all'esterno dell'astronave, nella sala di controllo del reattore.

I personaggi non compariranno, invece, nei seguenti luoghi: (ufficio,negozio, banca,cinema, stadio, mostra d'arte, chiesa), in quanto sono luoghi già dotati di una loro efficienza; inoltre essi non saranno presenti neanche all' esterno dell' astronave a prua, giacché in questi luoghi è presente il secondo pilota e uno potrebbe chiedersi perché il personaggio è con il secondo pilota e non gli presta aiuto.

Si è deciso di impostare un limite massimo di 100 personaggi per questa modifica, anche se la storia prevede un equipaggio di 250 passeggeri.

Il giocatore potrà comunicare con il personaggio attraverso due comandi appositi:

comunica e interagisci.

Inoltre Il gioco deve dirigere l'interazione del giocatore con i personaggi, andando ad aggiungere un' azione apposita nel gioco.

Dovrà essere amministrata la comparsa e la scomparsa dei personaggi dalle stanze all'inizio e alla fine della mossa.

Per far fronte a questo problema, all'inizio della mossa verrà controllato il luogo attuale, se è diverso da quello precedente allora verrà selezionato un personaggio, e verrà indicato che il personaggio è presente.

Alla fine della mossa viene controllato se il personaggio è presente, se questo è presente allora viene rimosso.

2.17 CHIESA

Nel progetto è richiesto l'ampliamento del mondo di gioco con l'aggiunta di un nuovo luogo identificato come "Chiesa". In questo luogo il giocatore potrà svolgere diverse attività più o meno utili ai fini della risoluzione del gioco.

1. Oggetti

Per poter svolgere le attività presenti nel nuovo luogo sono stati pensati dei nuovi oggetti che andranno a popolarlo.

2. Croce

Nel nuovo luogo "Chiesa" è stato inserito un nuovo oggetto denominato "Croce". Esso potrà essere preso dal personaggio e successivamente se lo ritroverà nell'inventario in quanto oggetto trasportabile.

3. Panchina

Nel nuovo luogo "Chiesa" è stato inserito un nuovo oggetto denominato "Panchina". Esso non potrà essere preso in quanto è un oggetto non trasportabile, ma sarà utile al personaggio per capire il comando da digitare per compiere una nuova azione (Esempio: vedendo appunto una panchina, al personaggio può venire in mente intuitivamente di digitare il comando "siediti" o "siediti panchina" e verrà attivata una nuova azione; vedi paragrafo 3.6.2).

4. Confessionale

Nel nuovo luogo "Chiesa" è stato inserito un nuovo oggetto denominato "Confessionale". Esso non potrà essere preso in quanto è un oggetto non trasportabile, ma anch'esso sarà utile al personaggio per capire il comando da digitare per compiere una nuova azione (Esempio: vedendo appunto un confessionale, al personaggio può venire in mente intuitivamente di digitare il comando "entra nel confessionale" o "confessati" e verrà attivata una nuova azione; vedi paragrafo 3.6.3).

5. Sacerdote

Nel nuovo luogo "Chiesa" è stato inserito un nuovo oggetto denominato "Sacerdote". Esso non potrà essere preso in quanto è un oggetto non trasportabile, ma sarà utile al personaggio per capire il comando da digitare per compiere una nuova azione (Esempio: vedendo appunto un sacerdote, al personaggio può venire in mente intuitivamente di digitare il comando "parla sacerdote" e verrà attivata una nuova azione; vedi paragrafo 3.6.4).

6. Azioni

Le azioni pensate per far interagire il personaggio con il nuovo luogo fanno riferimento a delle attività che di solito vengono svolte nel luogo Chiesa e agli oggetti precedentemente elencati nel paragrafo 2.2.

Il personaggio potrà quindi:

- prendere l'oggetto croce, che ai fini del gioco risulterà inutile ma comporterà una perdita di tempo (unità di vita);
- sedersi sull'oggetto panchina per riposarsi e recuperare delle unità di salute (in particolare 10), qualora durante il gioco avesse contratto delle ferite, pur non curandosi nel luogo Pronto Soccorso. Infatti se un personaggio contrae delle ferite, le unità di salute vengono incrementate ogni mossa fino all'azzeramento e quindi la fine della partita. Questo recupero di salute avverrà solo la prima volta che ci si siede e il suo scopo quindi è quello di dare al personaggio la possibilità di fare qualche mossa in più prima che la salute si azzeri completamente. Le ferite contratte precedentemente dal giocatore però rimarranno e quindi il decremento delle unità di vita continuerà normalmente.

Se il giocatore dovesse uscire dal luogo chiesa e rientrare successivamente potrà nuovamente sedersi una o più volte ma come prima, solo la prima volta potrà recuperare le unità di salute; questo può essere vantaggioso per il personaggio in quanto recupera unità di salute, ma è anche svantaggioso in quanto si perderà del tempo (unità di vita), quindi starà al giocatore prendere la giusta decisione.

- confessarsi nell'oggetto confessionale: quest'azione sarà una perdita di tempo in quanto ai fini del gioco risulterà inutile.

Per riprodurre la presenza di una fila di persone in attesa di poter utilizzare il "Confessionale" a ogni ingresso, si genera casualmente un valore T compreso in un intervallo definito tra 0 e 10. Prima di poter usufruire del "Confessionale", il giocatore può decidere se attendere per un tempo T o abbandonarlo. Qualora dovesse attendere, il personaggio dovrà subire una perdita di tempo (unità di vita) aggiuntiva pari a T, in caso contrario non ci sarà questo ulteriore decremento di tempo.

- parlare con l'oggetto sacerdote che risulterà utile in quanto esso fornirà un importante suggerimento; verrà cioè svelata la possibilità di svolgere una nuova azione, ossia pregare, sempre all'interno di questo luogo e che risulterà importante ai fini del gioco.
- pregare: quest'attività potrà essere svolta infinite volte ma con effetti differenti; infatti si genera casualmente un valore T compreso in un intervallo definito tra 0 e 10 e se quest'ultimo sarà maggiore di 5, non ci sarà nessun effetto, ma solo un messaggio a video con conferma di aver pregato. Al contrario invece, se questo sarà minore di 5, avverrà un miracolo al personaggio qualora però durante lo svolgimento del gioco avesse contratto una o più ferite.

Il personaggio verrà quindi guarito da una delle sue ferite e il suo stato salute tornerà a 100 (se non dovesse avere ulteriore ferite) oppure a $90+X$, dove X sarà un numero casuale compreso in un intervallo tra 0 e 9.

Il degrado della salute incide direttamente sul tempo necessario per svolgere le azioni offerte dal gioco: infatti un qualsiasi stato di degrado aumenterà il tempo utile nella realizzazione di qualsiasi azione all'interno dell'avventura. Le ferite hanno un effetto diretto sullo Stato di Salute peggiorandolo nel corso del tempo.

2.18 UFFICIO / DEPOSITO

Nel progetto è stata ripristinata la funzionalità Ufficio / deposito.

Questo luogo è stato realizzato per concedere al giocatore la possibilità di consegnare e ritirare documenti.

Nel luogo sono presenti un calcolatore da dove è possibile ottenere l'elenco dei documenti e una scrivania, dove invece possono essere consegnati i documenti.

Una volta preso un solo documento da un luogo specifico (incluso l'Ufficio / deposito stesso), il giocatore potrà consultarlo in qualsiasi momento e in qualsiasi luogo.

Ogni documento è composto da:

- Un nome che permette di identificare il documento nell'inventario del giocatore o quando viene stampata dal calcolatore la lista dei documenti disponibili per il ritiro;

2.19 SALI&TABACCHI

Il luogo che è stato implementato è il “Sali&Tabacchi”. Esso non ha finalità per la risoluzione del gioco, ma ha l'obiettivo di:

- Intrattenere il giocatore.
- Distrarlo dalla sua missione di salvataggio.
- Permettergli di incrementare tempo e denaro solo al verificarsi di determinate condizioni.

All'interno del luogo si troveranno oggetti come:

- Sale
- Sigarette
- Lotteria
- Elaboratore bonus

2.20 DISTANZA/SFORZO

Nel progetto è stato richiesto di integrare la funzionalità “distanza/sforzo” presente nel progetto di Andrea De Rinaldis.

Tale funzionalità dovrà garantire il calcolo automatico degli spostamenti effettuati dal giocatore all'interno della mappa di gioco, definendo uno sforzo fisico legato alla distanza effettuata e al consumo di energie.

Tuttavia, l'utente potrà decidere di visualizzare un certo numero di spostamenti effettuati, osservandone il percorso e l'ordine dei luoghi visitati.

Infine, l'utente potrà decidere di azzerare i calcoli effettuati, sia per la distanza percorsa che per lo sforzo fisico, determinando l'inizio di un nuovo conteggio.

2.21 FABBRICA

In base alla traccia precedentemente citata, è stato aggiunto un luogo “Fabbrica” nel quale il protagonista potrà parlare con il fabbricante, unico oggetto inserito all’interno del luogo, che gli chiederà di scegliere quale oggetto voglia costruire. Una volta effettuata la scelta il protagonista sceglierà uno alla volta il pezzo necessario, cercando di rispettare la sequenza di costruzione. Una volta che la costruzione è terminata con successo viene aggiunto del denaro al portafoglio del protagonista, che varia in base a cosa è stato precedentemente scelto di costruire.

2.22 TELETRASPORTA

Il progetto consiste nell’integrare una funzione di teletrasporto che permetterà al giocatore di muoversi verso un luogo da lui indicato, specificandone il suo numero identificativo. Per utilizzare questa funzione il giocatore dovrà far uso del comando “**teletrasporta**”, che sarà sfruttabile in ogni momento del gioco, al seguito del quale gli verrà richiesto di inserire il codice univoco del luogo verso cui egli vuole andare, a patto che sia valido e che non identifichi il luogo in cui il giocatore si trova in quell’istante. Inoltre, potrà essere usata, in modo iterativo, la parola indietro, che permetterà di ritornare automaticamente nell’ultimo luogo in cui il teletrasporto è stato usato (l’iterazione è possibile fino a quando non si arriva alla posizione di partenza, ovvero la prima stanza in cui la funzione è stata utilizzata).

2.23 CASSAFORTE E CASSETTA DI SICUREZZA MALMESSA

Il progetto riguarda l’integrazione del luogo *Cassaforte*, a sud della *Banca*, all’interno del quale il protagonista potrà interagire con una cassetta di sicurezza malmessa, unico oggetto qui presente. Questa cassetta non fornisce aiuti né è un oggetto chiave che permette al giocatore di migliorare il suo stato di avanzamento nel gioco, ma fornisce un semplice inventario “di riserva” nel quale il giocatore può depositare i propri oggetti o ritirarli, in qualunque momento, fino ad un massimo di dieci.

2.24 VERBO "RITORNA" E VERBO "RESOCONTO"

Il progetto riguarda l’integrazione della funzionalità “ritorna” che consente al giocatore di ripercorrere a ritroso il percorso effettuato, ripercorrendo un luogo alla volta. Viene inoltre aggiunta la funzionalità “resoconto” che consente al giocatore di visionare, in qualsiasi momento e qualora lo voglia, l’elenco dei luoghi visitati fino a quel momento, compreso il luogo in cui si trova attualmente.

2.25 LABORATORIO ANALISI

La funzionalità inserita da Bottiglione (che a sua volta ha apportato modifiche al laboratorio di analisi introdotto da Scarci) prevede la possibilità, una volta raggiunto il laboratorio di analisi, di utilizzare un droide medico che simula le analisi di diabete e/o colesterolo, generando un numero random all’interno di un range numerico e, sulla base del valore generato, comunicare all’utente se il livello di glicemia/colesterolo è nella norma oppure troppo alto (o troppo basso). Il dialogo con il droide viene implementato utilizzando un blocco switch che propone all’utente un codice diverso per ogni analisi da effettuare (1 per diabete, 2 per colesterolo, 3 per uscire) e, in base alla scelta effettuata, propone il risultato. Se si utilizza un comando diverso da 1,2 o 3, il programma risponde comunicando che la scelta effettuata non è valida.

2.26 GUARDA MAPPA

Il progetto consiste nell'integrare una funzione di visualizzazione mappa che permetterà al giocatore di accedere ad un menù ogni volta che ritiene necessario, con lo scopo di aprire una mappa che lo aiuterà nei suoi spostamenti in giro per il gioco. Questa funzione sarà attivabile in ogni momento della partita e in ogni luogo della mappa.

2.27 CUCINA

Il progetto consiste nell'integrare un luogo "Cucina" in cui il giocatore potrà placare l'appetito e recuperare salute mangiando cibi vari già presenti nel frigorifero o preparabili attraverso l'utilizzo dell'oggetto Bimby. La stanza sarà, inoltre, dotata di una televisione che si può accendere e guardare.

2.28 CAPACITÀ

Le capacità, che dovranno essere integrate nel progetto base, aumentano il livello di difficoltà, e ognuna di queste è un'azione ricorrente nel gioco:

- Aprire
- Digitare
- Leggere
- Scrivere
- Piegarsi
- Premere
- Saltare
- Guidare

Di queste, le prime quattro sono base, ossia sono presenti fin dall'inizio del gioco, mentre le altre si ottengono effettuando particolari azioni. Se il giocatore ha intenzione di premere un pulsante ma non ha la corrispondente capacità, allora apparirà un messaggio di errore.

È possibile ricordare solo cinque capacità alla volta, per cui bisognerà eliminarne qualcuna al momento adeguato. Per fare ciò, bisogna digitare `elimina capacita'`. Per sapere quali capacità sono attualmente in possesso si deve digitare `elenca capacita'`.

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.

3.1.1Struttura dati

La funzionalità *Percorsi museo* permette al protagonista di spostarsi tra le gallerie e le stanze del museo visionando le opere d'arte.

Questa funzionalità utilizza la struttura dati Grafo dove ogni nodo rappresenta una stanza del museo mentre ogni arco rappresenta la possibilità di spostarsi da una stanza all'altra.

La struttura dati **Grafo** è idonea perché simula i reali percorsi all'interno dei Musei, come per i grafi anche il protagonista può visitare solo le stanze adiacenti alla sua.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Museo.

3.2 SCUOLA E AULE

3.2.1Luogo 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".

3.2.2Luogo 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.

3.2.3Luogo 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.

3.2.5 Struttura dati

Nei luoghi Scuola, Classe 1A, Classe 2A, Classe 3A ci sono diverse funzionalità (es. risolvere cruciverba o giocare all’impiccato) ma tutte sono gestite senza l’ausilio di strutture dati.

Viene utilizzata la struttura dati Dizionario per l’inserimento dei vocaboli del luogo Scuola.

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 applicazione 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 ritornerà 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.

3.3.4 Struttura dati

Nel luogo Archivio ci sono alcune funzionalità (es. utilizzare il computer o leggere i files dopo averli scaricati) ma tutte sono gestite senza l'ausilio di strutture dati.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Archivio.

Sarebbe stato opportuno utilizzare la struttura dati Coda per simulare la fila delle persone in coda per usare il computer; inoltre sarebbe stato opportuno utilizzare la struttura Lista per gestire i files dell'Archivio.

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.

3.4.6 Struttura dati

La funzionalità Leggi libri permette di memorizzare i libri letti presenti in Aula.

Questa funzionalità utilizza la struttura dati Lista dove vengono inseriti al suo interno i libri letti dal protagonista.

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.

Dato che i libri da leggere sono un numero fisso e non è possibile leggere più volte lo stesso libro viene meno la necessità di utilizzare una struttura dati dinamica ma si potrebbe usare semplicemente una struttura dati a dimensione fissa come il vettore.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Aula.

3.5.1 Struttura dati

Nel luogo Ufficio postale ci sono diverse funzionalità (es. invio/ricezione di messaggi e pacchi) ma tutte sono gestite senza l'ausilio di strutture dati.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Ufficio postale.

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

2.1. 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

3.1. 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.

3.6.1 Struttura dati

Nel luogo Ufficio postale ci sono diverse funzionalità (es. invio/ricezione di messaggi e pacchi) ma tutte sono gestite senza l'ausilio di strutture dati.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Ufficio postale.

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.

3.7.4 Struttura dati

Nel luogo Sala scommesse non è presente alcuna funzionalità e alcun oggetto ma è presente solo il personaggio Joe con cui il protagonista non può interagire.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Sala Scommesse.

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 se 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).

3.8.1 Struttura dati

La funzionalità Jukebox permette al protagonista di scegliere le canzoni da ascoltare da un elenco predefinito.

Questa funzionalità utilizza la struttura dati Lista attraverso la quale il protagonista riesce a leggere le informazioni del brano in esecuzione e può scegliere un altro brano.

La struttura dati Lista è idonea perché simile alla struttura lineare del jukebox, inoltre gli operatori successivo e precedente simulano lo spostamento in "avanti" ed in "indietro" dei brani.

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.

3.9.1 Struttura dati

La funzionalità Biglietto permette di gestire i biglietti.

Questa funzionalità utilizza la struttura dati Dizionario formato dalla coppia <chiave, valore>, dove la chiave è il codice (univoco) del luogo (un intero), il valore è rappresentato dalla classe Biglietto la quale formata dalla coppia nome della destinazione e costo.

La struttura dati Dizionario è idonea perché la massima esigenza è quella di ritrovare in modo efficiente le destinazioni, invece l'inserimento viene effettuato poche volte.

La funzionalità Fila biglietteria rende più realistica l'esperienza dell'acquisto del biglietto da parte del protagonista.

Questa funzionalità utilizza la struttura dati Coda per gestire la fila di persone in coda alla biglietteria della Stazione per acquistare il biglietto del treno.

La struttura dati Coda è idonea perché simula il reale scorrimento della fila delle persone, il protagonista viene aggiunto ad un estremo (fondo) e le persone acquistano il biglietto dall'altro estremo (testa).

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Stazione.

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.

3.10.1 Struttura dati

La funzionalità Crea scheda permette di creare una nuova scheda di allenamento per il protagonista.

Questa funzionalità utilizza due strutture dati: la Lista che contiene i soli nomi delle schede, ed è utilizzata per mostrare a video le schede esistenti; il Dizionario contiene le schede sotto forma di coppie chiave-valore dove la chiave è la stringa contenente il nome della scheda ed il valore è il contenuto informativo della scheda stessa.

Quando si tenta di inserire una nuova scheda viene effettuato un controllo sui nomi già presenti nella lista e, nel caso si trovasse un “doppione”, questo verrà cancellato e di conseguenza la lista schede non può avere occorrenze.

La struttura dati Lista è idonea perché tiene conto della posizione delle schede dei clienti e permette la stampa dei nomi delle schede; la struttura dati Dizionario è idonea perché la massima esigenza è quella di ritrovare in modo efficiente gli esercizi della scheda.

La funzionalità Usa panca – Usa tapis permette al protagonista di usare gli attrezzi di allenamento della palestra.

Questa funzionalità utilizza la struttura dati Coda per gestire la fila di clienti in coda all’attrezzo selezionato.

La struttura dati Coda è idonea perché simula il reale scorimento della fila dei clienti, il protagonista viene aggiunto ad un estremo (fondo) e il cliente usa l’attrezzo dall’altro estremo (testa).

Viene utilizzata la struttura dati Dizionario per l’inserimento dei vocaboli del luogo Palestra.

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.

3.11.2 Struttura dati

La funzionalità Slot machine consente al protagonista di tentare la fortuna giocando alla slot machine della Sala Giochi.

Questa funzionalità utilizza la struttura dati Insieme che contiene gli eventi favorevoli e non che permetteranno di vincere o perdere.

La struttura dati Insieme è idonea perché non si tiene conto della posizione degli eventi ed ogni evento non può essere ripetuto più volte.

Viene utilizzata la struttura dati Dizionario per l’inserimento dei vocaboli del luogo Sala Giochi.

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.

3.12.1 Struttura dati

La funzionalità Libri in prestito permette di memorizzare i libri presi in prestito dalla Biblioteca. La struttura dati utilizzata è la Lista, questa struttura è idonea perché tiene conto della posizione dei libri in prestito e gestisce correttamente le occorrenze (anche se nella vetrina non ci sono libri uguali).

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Biblioteca.

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.

3.13.1 Struttura dati

La funzionalità Autobus consente al protagonista di utilizzare il mezzo di trasporto per raggiungere più velocemente determinati luoghi. È possibile accedere all'autobus solo dopo aver preso il ticket bus.

Questa funzionalità è gestita senza l'ausilio di strutture dati.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli per l'Autobus.

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.

3.14.1 Struttura dati

La funzionalità Visualizza persone in attesa permette al protagonista di visualizzare l'elenco dei clienti in attesa, mentre la funzionalità prenotati permette al protagonista di prenotarsi per il cambio batteria dell'automobile. In entrambe le funzionalità viene utilizzata la struttura dati Coda.

Questa struttura è idonea perché consente di aggiungere i clienti ad un estremo (fondo) e leggere i clienti dall'altro estremo (testa).

La funzionalità Gestione batterie permette al protagonista di visionare la pila di batterie e scartare quelle scariche. La struttura dati utilizzata è la Pila, questa struttura è idonea perché consente di aggiungere e rimuovere le batterie solo da un estremo (testa).

La funzionalità Diario meccanico permette al protagonista di visualizzare l'elenco delle attività del meccanico. La struttura dati utilizzata è la Lista, questa struttura è idonea perché simile alla struttura lineare del diario, inoltre gli operatori successivo e precedente simulano lo sfogliare delle pagine.

Viene utilizzata la struttura dati Dizionario per l'inserimento del personaggio Meccanico e degli oggetti del luogo Meccanico.

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

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.

Nel progetto verrà anche aggiunta una cassaforte nella caserma in cui il giocatore potrà rubare di nuovo gli oggetti sequestrati dal carabiniere una volta aperta. Per aprire la cassaforte si userà una combinazione di 4 cifre numeriche, la combinazione è 2021 e sarà scritta nel messaggio stampato (camuffato) quando si vogliono vedere le armi nella vetrina.

Sempre nella caserma verrà aggiunto un elenco di armi inutilizzabili che il giocatore potrà visualizzare attraverso una vetrina.

3.15.1 Struttura dati

La funzionalità Ruba permette al protagonista di rubare determinati oggetti in determinati luoghi del gioco e conservarli in un inventario degli oggetti rubati. Viene utilizzata la struttura dati Dizionario per l'inserimento dei verbi ruba e rubati.

La funzionalità Perquisizione permette al personaggio Carabiniere di perquisire l'inventario degli oggetti rubati del protagonista. Viene utilizzata la struttura dati Dizionario per l'inserimento del personaggio Carabiniere.

La struttura dati utilizzata per queste funzionalità è idonea perché la massima esigenza è quella di ritrovare in modo efficiente l'elemento interessato.

La cassaforte per tenere traccia degli oggetti sequestrati farà uso della Struttura Dati Pila. Le armi che saranno elencate verranno rappresentate con una Struttura Dati Lista

3.16 RISTORANTE E FAST FOOD

Il livello di sazietà del comandante potrà essere controllato in qualsiasi momento della partita, attraverso un apposito comando che, oltre visualizzarne il valore, ne fornirà una valutazione compresa tra “Ottimo” e “Molto Critico”. Lo stato del personaggio verrà mostrato in due modi differenti:

- Tramite un messaggio che conterrà i valori dell'appetito, indicando con una valutazione la criticità di tale parametro;
- Tramite un avvertimento che conterrà “il pensiero” del personaggio, in modo da aiutare il giocatore a capire il tempo a disposizione per placare la fame del comandante.

Ogni avvertimento avrà un messaggio diverso in base all'appetito. Ogni azione compiuta farà decrescere il livello di sazietà, poiché comporterà il consumo di un quantitativo di energia. Inoltre, il tempo trascorso senza mangiare inciderà ulteriormente sull'appetito del comandante: il fattore di sazietà crollerà vertiginosamente ad ogni azione, se non si soddisferà l'appetito del personaggio dopo un lungo periodo e, insieme ad esso, anche il tempo a disposizione per terminare il gioco, diminuirà sempre più velocemente. Diventerà quindi di fondamentale importanza controllare lo stato di sazietà e reperire del cibo per alleviare la fame.

Quest'ultimo potrà essere reperito attraverso due luoghi di ristorazione, implementati in due diverse stanze dell'astronave non utilizzate, alle quali vi si potrà accedere tramite il corridoio principale. Nel gioco saranno quindi inseriti:

- Un fast food, luogo in cui si potrà ottenere del cibo meno nutriente, ma più facilmente acquistabile.
- Un ristorante, dove invece si potrà ottenere del cibo molto nutriente, ma acquistabile meno facilmente.

In ogni zona ristoro saranno presenti pasti di varie tipologie aventi un valore energetico differente indicante l'incremento del livello di sazietà del personaggio, ed una categoria di appartenenza, utile a far capire al giocatore il metodo per acquistare una determinata pietanza, tramite i “Buoni Pasto”.

Quest'ultimi saranno presenti in alcune zone della mappa, a partire dalla cabina del capitano, e assumeranno nomi differenti in base alla zona in cui sono posti. In assenza di essi, il giocatore non potrà acquistare cibo. Inoltre saranno in numero inferiore rispetto alle pietanze presenti nel gioco, e in luoghi non direttamente accessibili. In questo modo, il giocatore dovrà mostrare la sua bravura nel cercarli e nell'utilizzarli, poiché dovrà scegliere il cibo da comprare e quando consumarlo.

I buoni pasto saranno suddivisi in categorie (blu, giallo, verde), che identificheranno le pietanze acquistabili, in modo da aumentare leggermente la difficoltà del gioco: infatti, a seconda della categoria di appartenenza, un buono pasto ha potere di acquisto su cibi più o meno nutrienti. Essi, una volta raccolti, potranno essere controllati tramite un apposito comando, che simulerà la tasca del capitano.

Il cibo acquistato dovrà essere riposto in un apposito zaino termico, strumento necessario al trasporto delle cibarie. Esso sarà disponibile già dall'inizio dell'avventura nella cabina del capitano e, trasportandolo, potrà essere consultabile in qualsiasi momento della partita. Non sarà possibile però, acquistare vivande senza di esso.

3.16.1 Struttura dati

La funzionalità Zaino termico consente al protagonista di prendere e conservare o mangiare il cibo acquistato al Ristorante o Fast Food.

Questa funzionalità utilizza la struttura dati Lista per gestire i piatti acquistati. Attraverso l'operatore di inserimento è possibile aggiungere nuovi piatti acquistati, con l'operatore di rimozione è possibile cancellare i piatti mangiati e con l'operatore di lettura è possibile visionare l'intero elenco dei piatti contenuti nello zaino.

La struttura dati Lista è idonea ma dato che non ci possono essere occorrenze di piatti si potrebbe usare anche la struttura dati Insieme.

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli dei luoghi Ristorante e Fast Food.

3.17 ESTRAZIONE DI UN PERSONAGGIO CASUALE

Come discusso in precedenza, la traccia richiedeva di aggiungere ad un progetto già esistente la classe Personaggi.

Per fare ciò si sono rese necessarie alcune modifiche, in particolare aggiunte di azioni metodi e oggetti.

Le modifiche che sono state effettuate hanno consentito il concretizzarsi delle seguenti azioni: inserire un personaggio, inserire le rispettive frasi che il personaggio dirà al protagonista, la possibilità del protagonista di invocare un personaggio e la gestione dei luoghi in cui il personaggio apparirà.

3.17.1. Personaggi

Il personaggio quindi, conterrà le seguenti informazioni:

nome del personaggio, le frasi che conosce, il luogo in cui è situato, se sono necessari degli oggetti per far cambiare interazione, e quali oggetti.

In particolare:

Il nome del personaggio sarà rappresentato da una stringa, il luogo in cui è situato il personaggio verrà rappresentato con un intero positivo, le frasi saranno rappresentate con delle stringhe. Nello specifico, l'informazione del luogo sarà indicata con -20, questo perché la funzione che gestisce i luoghi è una funzione apposita, che non ha niente a che vedere con la funzione principale che gestisce gli oggetti.

Il protagonista potrà colloquiare con il personaggio tramite dei comandi, che saranno nella forma oggetto+verbo, questi saranno gestiti dal sistema, attraverso dei codici assegnati loro.

Per precisione, ci sarà un codice formato da 4 cifre, le prime due che servono ad identificare il verbo, e le restanti due che servono ad identificare l'oggetto.

Ad esempio, nel comando "Comunica con Gabriel" avremo un codice 3973, 39 rivolto al verbo "comunica" e 73 rivolto all'oggetto (personaggio) "Gabriel".

La congiunzione "con", viene trascurata, grazie al fatto che a tutte le congiunzioni e/o articoli è stato assegnato un codice, nel nostro caso 7, e quindi il sistema quando incontrerà il valore 7 ignorerà quel termine

3.18 CHIESA

3.18.1. Inserimento del luogo "Chiesa" nel gioco

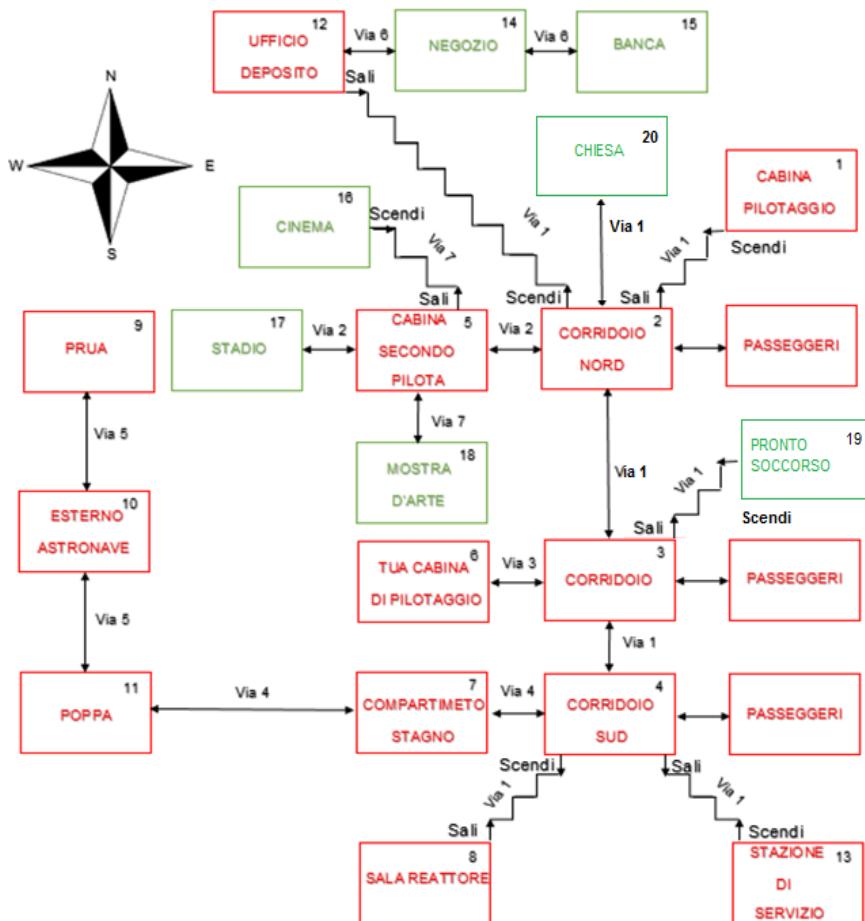
Come illustrato nel paragrafo 2.1 della Documentazione, è stato implementato il nuovo luogo "Chiesa" per permettere al Giocatore di svolgere delle nuove attività. Il luogo è identificato da:

- Un codice identificativo univoco di tipo numerico che lo distingue dagli altri luoghi presenti nel Gioco;
- Un codice di tipo numerico che identifica i percorsi utilizzabili all'interno del luogo. Tale codice è formato da dodici cifre aventi il seguente significato: NNSSEEOOUUDD dove N=Nord, S=Sud, E=Est, O=Ovest, U=Sali, D=Scendi;
- Un'etichetta di tipo testuale che identifica il nome del luogo.

Codice Mappa	Codice Luoghi	Nome
20	000200000000	Chiesa

Inoltre, per creare un senso di coerenza all'interno della Mappa, si è deciso di inserire il luogo "Chiesa" sopra il luogo "Corridoio Nord". Pertanto, scendendo dal luogo "Chiesa" sarà possibile tornare all'interno del "Corridoio Nord".

Qui di seguito è riportata la mappa aggiornata.



3.18.2. Inserimento "Croce" nel gioco

Come descritto nel paragrafo 2.2.1 della documentazione. Il progetto richiede l'integrazione di tale oggetto nel mondo di gioco nella stanza "Chiesa". L'oggetto si troverà in questo luogo e potrà essere raccolto senza particolari requisiti. In definitiva è stato aggiunto agli oggetti di gioco con delle caratteristiche univoche come l'etichetta e il suo codice univoco per non creare conflitti e come ultimo parametro la stanza in cui è possibile procurarla. Il codice mappa sarà positivo per indicare la possibilità per il personaggio di prendere l'oggetto (trasportabile).

ETICHETTA	CODICE OGGETTO	CODICE MAPPA
<i>Una croce</i>	43	20

Dopo averlo aggiunto agli oggetti di gioco è necessario anche inserire la Croce tra i vocaboli accettati all'interno del gioco, con una nuova etichetta che sarà la parola chiave da utilizzare per poter interagire con l'oggetto e il codice oggetto definito in precedenza. Una possibile rappresentazione di ciò potrebbe essere:

ETICHETTA	CODICE OGGETTO
<i>Croce</i>	43

3.18.3. Inserimento "Panchina" nel gioco

Come descritto nel paragrafo 2.2.2 della documentazione. Il progetto richiede l'integrazione di tale oggetto nel mondo di gioco nella stanza "Chiesa". L'oggetto si troverà in questo luogo e non potrà essere preso. In definitiva è stato aggiunto agli oggetti di gioco con delle caratteristiche univoche come l'etichetta e il suo codice univoco per non creare conflitti e come ultimo parametro la stanza in cui è possibile procurarla. Il codice mappa sarà negativo per indicare l'impossibilità per il personaggio di prendere l'oggetto (non trasportabile).

ETICHETTA	CODICE OGGETTO	CODICE MAPPA
<i>Panchina</i>	92	-20

Dopo averlo aggiunto agli oggetti di gioco è necessario anche inserire il gettone tra i vocaboli accettati all'interno del gioco, con una nuova etichetta che sarà la parola chiave da utilizzare per poter interagire con l'oggetto e il codice oggetto definito in precedenza. Una possibile rappresentazione di ciò potrebbe essere:

ETICHETTA	CODICE OGGETTO
<i>Panchina</i>	92

3.18.4. Inserimento "Confessionale" nel gioco

Come descritto nel paragrafo 2.2.3 della documentazione. Il progetto richiede l'integrazione di tale oggetto nel mondo di gioco nella stanza "Chiesa". L'oggetto si troverà in questo luogo e non potrà essere trasportato. In definitiva è stato aggiunto agli oggetti di gioco con delle caratteristiche univoche come l'etichetta e il suo codice univoco per creare conflitti e come ultimo parametro la stanza in cui è possibile procurarla e definito come oggetto non trasportabile.

ETICHETTA	CODICE OGGETTO	CODICE MAPPA
<i>Un confessionale</i>	97	-19

Dopo averlo aggiunto agli oggetti di gioco è necessario anche inserire il terminale tra i vocaboli accettati all'interno del gioco, con una nuova etichetta che sarà la parola chiave da utilizzare per poter interagire con l'oggetto e il codice oggetto definito in precedenza. Una possibile rappresentazione di ciò potrebbe essere:

ETICHETTA	CODICE OGGETTO
<i>Confessionale</i>	97

3.18.5. Inserimento "Sacerdote" nel gioco

Come descritto nel paragrafo 2.2.4 della documentazione. Il progetto richiede l'integrazione di tale oggetto nel mondo di gioco nella stanza "Chiesa". L' oggetto si troverà in questo luogo e non potrà essere trasportato. In definitiva è stato aggiunto agli oggetti di gioco con delle caratteristiche univoche come l'etichetta e il suo codice univoco per con creare conflitti e come ultimo parametro la stanza in cui è possibile procurarla e definito come oggetto non trasportabile.

ETICHETTA	CODICE OGGETTO	CODICE MAPPA
<i>Un sacerdote</i>	98	-19

Dopo averlo aggiunto agli oggetti di gioco è necessario anche inserire il terminale tra i vocaboli accettati all'interno del gioco, con una nuova etichetta che sarà la parola chiave da utilizzare per poter interagire con l'oggetto e il codice oggetto definito in precedenza. Una possibile rappresentazione di ciò potrebbe essere:

ETICHETTA	CODICE OGGETTO
<i>Sacerdote</i>	98

3.18.6. Inserimento delle Azioni

Come descritto nel paragrafo 2.3 della documentazione, le "Azioni" sono una delle parti fondamentali del progetto. Il giocatore potrà entrare in qualsiasi momento nel nuovo luogo "Chiesa" e compiere una o più attività in esso presenti.

3.18.6.1. Inserimento Azione "Prendi croce" nel gioco

Come anticipato nel paragrafo 3.2 della Documentazione, si è deciso di inserire l'Oggetto trasportabile "Croce" all'interno del nuovo luogo "Chiesa".

L'azione "Prendi croce" permetterà al giocatore di usufruire del servizio offerto dalla "Chiesa".

Prendere l'oggetto croce ai fini del gioco risulterà inutile ma comporterà una perdita di tempo (unità di vita).

Impostando il codice dell'oggetto "croce" come trasportabile, esso potrà essere preso in qualsiasi momento e il personaggio lo avrà sempre con se nell'inventario.

3.18.6.2. Inserimento Azione "Siediti" nel gioco

Come anticipato nel paragrafo 3.3 della Documentazione, si è deciso di inserire l'Oggetto non trasportabile "Panchina" all'interno del nuovo luogo "Chiesa". Poiché l'oggetto permetterà una interazione con il Giocatore, si è deciso di aggiungere la nuova azione "Siediti" e un suo sinonimo, "Siediti Panchina".

Queste azioni permetteranno al giocatore di usufruire dei servizi offerti dalla "Chiesa".

L'azione "Siediti" sarà aggiunta alla tabella dei Vocaboli e sarà caratterizzata da:

- Un'etichetta di tipo testuale;
- Un codice identificativo univoco di tipo numerico che distinguerà il nuovo Vocabolo dagli altri presenti nella Tabella.

Etichetta	Codice Identificativo
<i>Siediti</i>	94

L'azione permetterà al personaggio di sedersi sull'oggetto panchina per riposarsi e recuperare delle unità di salute (in particolare 10), qualora durante il gioco avesse contratto delle ferite, pur non curandosi nel luogo Pronto Soccorso. Questo recupero di salute avverrà solo la prima volta che ci si siede e il suo scopo è quello di dare al personaggio, anche ferito, la possibilità di fare qualche mossa in più prima che la salute si azzeri completamente.

Se il giocatore dovesse uscire dal luogo chiesa e rientrare successivamente potrà nuovamente sedersi una o più volte ma come prima, solo la prima volta potrà recuperare le unità di salute; questo però comunque svantaggioso in quanto si perderà del tempo.

Per permettere ciò è stata aggiunta una nuova Azione di supporto che cambi il valore di una variabile di controllo non appena si esca dal luogo Chiesa. Essa non sarà visibile al giocatore, ma si attiverà non appena il giocatore digitò il comando "Sud", in quanto questa direzione è l'unica consentita dal luogo Chiesa verso gli altri.

3.18.6.3. Inserimento Azione "Confessati" nel gioco

Come anticipato nel paragrafo 3.4 della Documentazione, si è deciso di inserire l'Oggetto non trasportabile "Confessionale" all'interno del nuovo luogo "Chiesa". Poiché l'oggetto permetterà una interazione con il Giocatore, si è deciso di aggiungere la nuova azione "Confessati" e un suo sinonimo, "Entra confessionale".

Queste azioni permetteranno al giocatore di usufruire dei servizi offerti dalla "Chiesa".

L'azione "Confessati" sarà aggiunta alla tabella dei Vocaboli e sarà caratterizzata da:

- Un'etichetta di tipo testuale;
- Un codice identificativo univoco di tipo numerico che distinguerà il nuovo Vocabolo dagli altri presenti nella Tabella.

Etichetta	Codice Identificativo
<i>Confessati</i>	96

L'azione permetterà al personaggio di confessarsi, ma sarà una perdita di tempo in quanto ai fini del gioco risulterà inutile.

Per simulare la presenza di una coda in attesa di utilizzare il "Confessionale" all'interno della "Chiesa", si genererà un valore intero casuale compreso tra 0 e 10 ($0 \leq x \leq 10$) non appena avviata l'azione "Entra Confessionale" o "Confessati". Tale valore sarà stampato a video nel seguente modo:

- Numero_Persone attendono di poter usare il terminale. Desideri aspettare? [si/no]

Numero_Persone rappresenta il valore generato in maniera casuale. Qualora Numero_Persone fosse uguale a 0, si è deciso di permettere l'accesso al "Confessionale" senza alcuna perdita di tempo aggiuntiva, quindi solo il normale decremento di un unità.

Invece, per simulare il tempo sprecato per visionare il "Confessionale" e la coda, il tempo diminuirà di 1 anche nel caso in cui il Personaggio decidesse di non attendere. In questo caso, il Giocatore tornerà nel luogo "Chiesa" e potrà decidere di interagire ancora con il "Confessionale".

Qualora, invece, decidesse di attendere il suo turno, il tempo verrà decrementato di Numero_Persone unità e infine sarà svolta l'azione.

3.18.6.4. Inserimento Azione "Parla sacerdote" nel gioco

Come anticipato nel paragrafo 3.5 della Documentazione, si è deciso di inserire l'Oggetto non trasportabile "Sacerdote" all'interno del nuovo luogo "Chiesa". Poiché l'oggetto permetterà una interazione con il Giocatore, si è deciso di aggiungere la nuova azione "Parla Sacerdote", Tale azione farà uso dell'azione "Parla" preesistente nel progetto.

L'azione permetterà di parlare al sacerdote, che risulterà utile in quanto esso fornirà un importante suggerimento; verrà cioè svelata la possibilità di svolgere una nuova azione, ossia "Pregare", sempre all'interno di questo luogo e che risulterà importante ai fini del gioco.

3.18.6.5. Inserimento Azione "Prega" nel gioco

Per permettere al giocatore di pregare nel luogo "Chiesa", è stata aggiunta l'azione "Prega". Tale azione sarà aggiunta alla tabella dei Vocaboli e sarà caratterizzata da:

- Un'etichetta di tipo testuale;
- Un codice identificativo univoco di tipo numerico che distinguerà il nuovo Vocabolo dagli altri presenti nella Tabella.

Etichetta	Codice Identificativo
Prega	95

L'azione potrà essere svolta infinite volte ma con effetti ogni volta differenti; infatti si genera casualmente un valore T compreso in un intervallo definito tra 0 e 10 e se quest'ultimo sarà maggiore di 5, non ci sarà nessun effetto ma solo una conferma di aver pregato.

Al contrario invece, se questo sarà minore di 5, avverrà un miracolo al personaggio qualora durante lo svolgimento del gioco avesse contratto una o più ferite. Il personaggio verrà quindi guarito da

una delle sue ferite e il suo stato salute tornerà a 100 (se non dovesse avere ulteriore ferite) oppure a $90+X$, dove X sarà un numero casuale compreso in un intervallo tra 0 e 9.

Il degrado della salute incide direttamente sul tempo necessario per svolgere le azioni offerte dal gioco: infatti un qualsiasi stato di degrado aumenterà il tempo utile nella realizzazione di qualsiasi azione all'interno dell'avventura. Le ferite hanno un effetto diretto sullo Stato di Salute peggiorandolo nel corso del tempo.

Questa azione utilizza le funzionalità del "Pronto Soccorso", in particolare lo "Stato Salute".

Esso rappresenta la condizione fisica del personaggio e viene influenzato dalle "Ferite" contratte nel corso dell'avventura. A sua volta influenza il numero di mosse necessarie al giocatore per compiere una qualunque azione e, se non curato, può portare a una fine anticipata dell'avventura. Pertanto, lo "Stato di Salute" sarà composto nel seguente modo:

- Un valore di tipo intero rappresentante lo "Stato di Salute" del Personaggio. Tale valore è definito all'interno dell'intervallo $0 \leq x \leq 100$ dove l'estremo superiore rappresenta lo "Stato di Salute" ottimale del personaggio mentre quello inferiore la sua condizione di dipartita;
- L'elenco delle "Ferite" contratte dal Giocatore durante il corso dell'avventura;
- Un valore di tipo intero rappresentante lo "Stato di Degrado" del Personaggio. Tale valore è definito all'interno dell'intervallo $0 \leq x \leq 4$ e indica il numero di mosse aggiuntive necessarie al Giocatore per svolgere una qualsiasi azione nel corso dell'avventura.

Sarà quindi necessario apportare delle modifiche alla specifica sintattica e semantica dello "Stato di Salute". Qui di seguito verranno riportate le specifiche con in rosso le modifiche aggiunte.

3.18.6.5.1. Modifica Specifica Sintattica Dello Stato Di Salute

Stato di Salute è un Tipo di Dato astratto che contiene le seguenti informazioni:

- Un intero numerico (Salute) che rappresenta lo Stato di Salute;
- Un intero numerico (PenalitaMosse) che rappresenta lo Stato di Degrado del Personaggio;
- Una lista di Ferite (Ferite) che rappresentano le Ferite contratte nel corso dell'avventura.

Tipi: StatoSalute, Ferita, Interi, Lista, booleani

Operatori:

CreaStatoFerita () → StatoFerita

Ferito (StatoFerita) → Booleano

Cura (StatoSalute) → StatoSalute

SetFerita (StatoSalute, Ferita) → StatoSalute

GetPenalita (StatoSalute) → Interi

GetStatoSalute (StatoSalute) → Interi

Get_Ferite (StatoSalute) → Lista

Aggiorna_Stato (StatoSalute) → StatoSalute

Ferita_Contratta (StatoSalute, Ferita) → Booleano
SetStatoSalute (StatoSalute) → StatoSalute
Miracolo(StatoSalute) → StatoSalute

3.18.6.5.2. Modifica Specifica Semantica Dello Stato Di Salute

Tipi:

- **StatoSalute.** Insieme delle triple (S, D, L) dove S è l'insieme dei valori interi di Salute, D è l'insieme dei valori interi di PenalitaMosse e L è la Lista delle Ferite;
- **Ferita.** Insieme delle quadruple (N, D, C, S) dove N è l'insieme delle stringhe Nome, D è l'insieme delle stringhe Descrizione, C è l'insieme degli interi Codice e S è l'insieme degli interi DannoSalute.

Operatori:

CreaStatoSalute () = S

Post: StatoSalute S creato con Salute = 100, PenalitaMosse = 0 e Ferite = <>

Ferito (S) = b

Post: b = true se Ferite ≠ <>, false altrimenti

Cura (S) = S'

Pre: Il Giocatore è Ferito

Post: StatoSalute S' aggiornato con Salute = 100, PenalitaMosse = 0 e Ferite = <>

SetFerita (S, F) = S'

Post: S' aggiornato con Ferite = <F, F1, F2, ..., Fn> con $0 \leq n \leq 10$

GetPenalita (S) = I

Post: I = PenalitaMosse

GetStatoSalute (S) = I

Post: I = Salute

Get_Ferite (S) = L

Post: L = Ferite

Aggiorna_Stato (S) = S'

Pre: Il Giocatore è Ferito

Post: S' aggiornato decrementando a Salute il DannoSalute di ogni Ferita contratta e calcolando la nuova PenalitaMosse

Ferita_Contratta (S, F) = B

Post: B = true se $F \in L$, con $L = \langle F1, F2, \dots, Fn \rangle$ lista delle Ferite contratte dal Giocatore, con $0 \leq n \leq 10$

SetStatoSalute (S) = S'

Post: S' = S

Miracolo (S) = S'

Pre: Il Giocatore è Ferito

**Post: StatoSalute S' aggiornato con Salute = 100, PenalitaMosse = 0 se Ferite = <>,
Salute= 90+X, PenalitaMosse = 1 altrimenti**

3.18.1 Struttura dati

La funzionalità Prega consente di guarire le ferite del protagonista procurate durante la partita.

Questa funzionalità utilizza la struttura dati Lista dove vengono inserite le ferite riportate dal protagonista ed attraverso la preghiera si verifica il miracolo che genera l'eliminazione della prima ferita e ripristina lo stato di salute iniziale.

La struttura dati Lista è idonea perché tiene conto delle posizioni delle ferite inserite, dato che viene curata sempre la prima ferita inserita potrebbe essere usata anche la struttura dati Coda avendo una disciplina di accesso FIFO (“First In First Out”).

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli del luogo Chiesa.

3.19 UFFICIO / DEPOSITO

Come richiesto dalla traccia è stata ripristinata la funzionalità di tale luogo traendo le informazioni dal progetto da integrare.

Per fare ciò è stato necessario effettuare alcune modifiche in modo da aggiungere vocaboli e azioni mancanti ed eliminare le collisioni con le altre funzionalità del gioco.

3.19.1 Documenti

Il nome e la descrizione del documento sono rappresentati da stringhe, il luogo in cui è situato il documento è rappresentato da un intero ed infine il codice del documento è rappresentato da un intero positivo.

Il giocatore potrà ritirare un solo documento alla volta, che verrà custodito all'interno del suo inventario.

Tale documento può essere letto in qualsiasi luogo ed in qualsiasi momento e può essere consegnato dal giocatore recandosi nell'Ufficio / deposito.

La descrizione del documento coincide con il contenuto del documento stesso. Tale descrizione fornirà indizi importanti al giocatore per risolvere l'Adventure.

Infine il codice del documento è univoco e permette di identificare il documento al momento del ritiro o della consegna.

3.19.2 Struttura dati

Viene utilizzata la struttura dati Dizionario per l'inserimento dei vocaboli *ritira*, *consegna*, *documento* e *documenti*.

I documenti disponibili per il ritiro sono invece rappresentati mediante la struttura Lista. Attraverso la scansione sequenziale della lista dei documenti viene stampato a video l'elenco dei documenti disponibili.

3.20 SALI&TABACCHI

Il luogo da implementare, "Sali&Tabacchi", è stato inserito a ovest dell' "Aula", la quale si trova a sud del corridoio.

Gli oggetti presenti nel nuovo luogo saranno:

- Sale: oggetto trasportabile.
- Sigarette: oggetto trasportabile.
- Lotteria: oggetto non trasportabile che simulando il gioco della lotteria, in caso di vittoria permette di aumentare il denaro presente nel portafoglio. Il costo di un biglietto sarà di €1 e anche in caso di perdita, esso potrà essere utilizzato un'altra volta nell'elaboratore bonus per tentare di vincere 20 secondi bonus.
- Elaboratore bonus: oggetto non trasportabile che permette di convertire le ricevute degli acquisti effettuati all'interno del luogo Sali&Tabacchi in tempo bonus: per gli scontrini il bonus sarà uguale alla somma spesa più un'unità casuale aggiuntiva, mentre per i biglietti acquistati alla lotteria vi è la possibilità di tentare ancora la fortuna verificando se il numero scelto darà la possibilità di acquisire secondi bonus.

Prima di poter effettuare le operazioni di acquisto è stata simulata una fila di persone in coda, perciò verrà chiesto al giocatore se attendere o meno il suo turno; nel caso di

risposta affermativa verrà sottratto del tempo uguale al numero di persone in fila. Inoltre il giocatore avrà la possibilità di acquistare più prodotti dello stesso tipo all'interno della stessa fila, ma sarà creato uno scontrino singolo per ogni prodotto acquistato in modo da avere la possibilità di ricevere un bonus maggiore nella conversione degli scontrini attraverso l'elaboratore bonus.

Per poter effettuare le operazioni di acquisto il giocatore dovrà avere con se il portafoglio con l'importo minimo d'acquisto che varia in base al prodotto, altrimenti il sistema non permetterà di comprare nulla.

Per poter ricavare bonus dagli scontrini bisogna aver effettuato almeno un acquisto, mentre per tentare ancora la fortuna giocando i biglietti della lotteria bisognerà averne acquistato almeno uno, anche nel caso in cui non vi sia stata alcuna vincita, altrimenti il sistema non permetterà di ottenere alcun bonus.

3.21 DISTANZA/SFORZO

La funzionalità di calcolo della distanza e dello sforzo fisico è determinata in modo automatico dal sistema di gioco.

Tuttavia, l'utente può decidere di visionare i dati da essa calcolati utilizzando il termine sforzo: ciò permette l'apertura dell'interfaccia di presentazione dei dati calcolati tramite la funzionalità stessa.

In un primo momento, viene mostrato a video un messaggio che permette di visualizzare il numero di passi percorsi (passo inteso come spostamento da una determinata area della mappa di gioco ad un'altra adiacente).

Un altro messaggio permette all'utente di poter inserire un valore inherente al numero di passi da visualizzare.

Su tale valore si effettua un'analisi di correttezza: se esso è minore o uguale a zero viene richiesto nuovamente l'inserimento, se invece è un numero positivo maggiore del numero di passi, si effettua un ricalcolo della richiesta e verranno mostrati tutti i passi compiuti.

In base al valore inserito vengono mostrati i passi compiuti: si tratta di una lista di spostamenti, che vanno dal più recente al meno recente, in cui si possono visualizzare delle informazioni secondo il seguente costrutto:

Sei andato da: area → area pernmetri

Oltre alle descrizioni inerenti alle aree di partenza e di arrivo, si può visualizzare il numero di metri percorsi per ogni passo effettuato: ogni collegamento tra le diverse aree contiene uno specifico peso (determinato già nel progetto base per il calcolo del tempo rimanente) che specifica la lunghezza dello spostamento compiuto.

Successivamente vengono mostrati due diversi messaggi: uno è dedicato al conteggio della distanza compiuta, l'altro allo sforzo complessivo.

Entrambi i messaggi fanno riferimento ai passi che si stanno analizzando.

Infine, viene presentato un ultimo messaggio che richiede all'utente la propria scelta riguardo al ripristino del calcolo della distanza e dello sforzo: una risposta affermativa determina l'azzeramento di tutti i contatori e la chiusura della schermata della distanza/sforzo, una risposta negativa permette unicamente di ritornare all'interfaccia principale lasciando inalterati i dati sui passi.

In ogni caso, dopo la chiusura della schermata della distanza/sforzo, all’utente vengono mostrati i passi attuali.

3.21.1 Struttura dati

Per la realizzazione della funzionalità di calcolo della distanza e dello sforzo si fa uso di due strutture dati specifiche: due Pile.

Una pila, denominata “*traccia_passi*”, si occupa della memorizzazione delle informazioni riguardo gli spostamenti, definendo l’area di partenza e l’area di arrivo.

Un’altra pila, denominata “*calcola*”, si occupa della memorizzazione delle informazioni riguardo la distanza dei passi percorsi tra le due aree.

Entrambe hanno una correlazione comune: l’*n*-esimo elemento della pila “*traccia_passi*” definisce una distanza tra le due aree definite in esso pari al valore dell’*n*-esimo elemento della pila “*calcola*”.

3.22 FABBRICA

All’interno del luogo “*Fabbrica*” è stato inserito un solo oggetto, non trasportabile, che è il *mercante*. Si potrà usare il comando “*parla mercante*” o “*parla con mercante*” per interagire con esso. Una volta inserito il comando, verranno elencati gli oggetti costruibili:

- *Collana*, che conterrà in ordine:
 - *filo*;
 - *perline*;
 - *gancio*;
- *Tavolo*, che conterrà in ordine:
 - *piano in legno*;
 - *gambe*;
 - *giuntura*;
 - *viti*;
- *Porta in metallo*, che conterrà in ordine:
 - *telaio*;
 - *pannello in metallo*;
 - *cerniera*;
 - *serratura*;
 - *maniglia*;
- *Computer*, che conterrà in ordine:

- scheda madre;*
 - processore;*
 - scheda video;*
 - ventola;*
 - case;*
 - periferiche;*
- *Missile*, che conterrà in ordine:
- motore;*
 - tubo di scarico;*
 - sensore infrarossi;*
 - giroscopio;*
 - sistema di volo;*
 - detonatore;*
 - carica esplosiva;*
 - scocca;*
 - giunture;*
 - alette;*

Il protagonista effettuerà la scelta inserendo il numero corrispondente che sarà mostrato nella stampa degli oggetti costruibili. Verrà inoltre mostrato il guadagno che si avrà per ogni oggetto:

Collana: 2 Euro; Tavolo: 10 Euro; Porta in metallo: 20 Euro;

Computer: 50 Euro; Missile: 100 Euro;

3.22.1 Struttura dati

Gli oggetti costruibili saranno implementati con una Coda, nelle quali saranno inseriti in ordine le parti necessarie alla costruzione, in modo che ogni volta in cui il protagonista inserisce una parte, questa venga confrontata con la testa della coda (che sarà per l'appunto il pezzo necessario in quel momento); una volta che il confronto è confermato, e quindi il protagonista inserisce la parte esatta, la testa della coda viene cancellata, facendo così subentrare quello che era il successivo.

3.23 TELETRASPORTA

La funzionalità del teletrasporto potrà essere utilizzata dal giocatore in qualunque stanza egli si trovi, a patto che venga inserito un luogo esistente e diverso da quello in cui si trova attualmente o venga inserita la parola “indietro” per tornare alla stanza in cui si è utilizzata precedentemente tale funzione. Il giocatore potrà inoltre scegliere di non utilizzare questa funzionalità continuando a spostarsi normalmente per la mappa tramite gli appositi comandi: N=“nord”, E=“est”, O=“ovest”, S=“sud”, “sali”, “scendi”.

3.23.1 Struttura dati

La funzionalità teletrasporto richiede l'utilizzo della struttura dati "Pila", questa consente il salvataggio delle stanze in cui il giocatore ha utilizzato la funzione sopracitata, al fine di poter ritornare, inserendo la parola "indietro", all'interno di esse, potendo andare dunque di stanza in stanza a ritroso fino ad arrivare alla posizione di partenza (ovvero la prima stanza in cui è stato usato il teletrasporto).

3.24 CASSAFORTE E CASSETTA DI SICUREZZA MALMESSA

All'interno del luogo Cassaforte è stato inserito un solo oggetto, non trasportabile, che è la cassetta di sicurezza malmessa. Si potranno utilizzare tre comandi per interagire con questo oggetto:

- deposita *<codice_oggetto>*
 - A condizione che l'oggetto specificato sia presente nell'inventario del giocatore, utilizzare questa azione rimuove l'oggetto dall'inventario e lo inserisce nella cassetta.
- controlla cassetta
 - Usare questa azione mostrerà a schermo il contenuto della cassetta di sicurezza.
- ritira *<codice_oggetto>*
 - Usare questa azione permette di ritirare l'oggetto depositato dalla cassetta. Se questo non vi è presente, verrà mostrato un messaggio di errore.

3.24.1 Struttura dati

La struttura dati utilizzata per implementare la cassetta è quella del vettore. Avendo un numero fisso ad indicare il numero massimo di oggetti inseribili questa è la soluzione migliore perché consente di avere l'accesso diretto a tutti gli elementi. Per la ricerca di un elemento, si ha sforzo minimo perché basta iterare tra tutti gli elementi. L'unica operazione più faticosa in termini di computazione è quella della rimozione di un elemento non in ultima posizione, visto che poi bisogna far scalare i successivi, ma è un prezzo accettabile considerati i vantaggi.

3.25 Verbo "ritorna" e verbo "resoconto"

La funzionalità ritorna potrà essere utilizzata dal giocatore in qualunque stanza si trovi, eccetto il luogo "La tua cabina" poiché è il punto di partenza. Qualora sia usata la funzionalità all'interno della cabina verrà restituito un messaggio che informa dell'impossibilità della cosa. "Ritorna", converte i comandi ricevuti per permettere il percorso a ritroso, quindi N->S, S->N, O->E e E->O ed è il giocatore stesso a decidere quando utilizzarla. Per quanto riguarda la funzionalità resoconto, anch'essa è utilizzabile in qualsiasi luogo e consente di visionare, sempre a scelta del giocatore tutti i luoghi visitati fino a quel momento dal più recente, al meno recente, luogo attuale compreso.

3.25.1 Struttura dati

Entrambe le funzionalità richiedono l'uso della struttura dati pila. La funzionalità ritorno, utilizzerà la pila per memorizzare i comandi ricevuti in modo che siano poi convertiti, consentendo così il ritorno al luogo precedente e la funzionalità resoconto la usa per mostrare al giocatore i luoghi dal più recente al meno recente.

3.26 LABORATORIO ANALISI

Il Laboratorio analisi è raggiungibile proseguendo verso est da "La tua cabina", salendo si arriva al pronto soccorso e poi ad est ci sarà il luogo aggiunto. Con l'oggetto droide medico, con il quale l'utente potrà interagire, si potrà scegliere di far eseguire una fra due azioni, quali: analisi colesterolo e analisi diabete. Verranno, quindi, calcolati in modo casuale o il colesterolo o il diabete e l'utente potrà conoscere tali valori con un messaggio stampato a video indicante il livello (sotto forma di numero intero) e, a seconda del livello, un messaggio di condizioni sulla gravità della situazione di salute.

3.26.1 Struttura dati

Le realizzazioni delle strutture dati presenti nel progetto di Bottiglione erano già presenti in maniera identica nel progetto base di Palemburgi, eccezion fatta per le realizzazioni di Coda con priorità con heap e Dizionario con array con liste di trabocco.

Per il primo caso – Coda con priorità con heap – utilizzo la realizzazione presente nel progetto base. L'unica differenza con la realizzazione presente nel progetto di Bottiglione, infatti, è la dichiarazione della lunghezza massima del vettore heap. Credo sia più corretto dichiarare la lunghezza come una costante piuttosto che come una semplice variabile intera. A livello funzionale non ci sono differenze. Per il secondo caso, invece, utilizzo la realizzazione di Bottiglione perché fa uso della parola chiave *const* (che garantisce maggiore sicurezza dei dati perché non possono essere modificati) e del riferimento & per garantire un minor spreco di memoria.

3.27 GUARDA MAPPA

La funzionalità "guarda mappa" potrà essere utilizzata dal giocatore in qualunque stanza del gioco.

Avrà a disposizione un menù dove potrà scegliere se visualizzare la mappa con i luoghi adiacenti a quello in cui si trova o di visualizzare l'intera mappa di gioco.

. Il giocatore potrà inoltre scegliere di non utilizzare questa funzionalità continuando a spostarsi normalmente per la mappa tramite gli appositi comandi: N="nord", E="est", O="ovest", S="sud", "sali", "scendi".

3.27.1 Struttura dati

La funzionalità guarda mappa richiede l'utilizzo della struttura dati "Grafo", questa consente di memorizzare le informazioni relative ai luoghi e ai collegamenti tra di essi.

3.28 CAPACITÀ

Il collega Vergine Erik aveva integrato le capacità fino all'azione *azione_138 (guarda computer)*. Oltre a utilizzare il suo codice per reintegrarle fino a dove era arrivato (integrandole modifiche effettuate dagli altri colleghi successivamente) bisogna aggiornare anche le azioni introdotte successivamente, verificando per ogni azione che richiede una (o più) capacità se essa sia presente nell'elenco di capacità apprese.

Inoltre, Vergine non ha fornito al giocatore una spiegazione di questa nuova funzionalità e soprattutto non ha inserito l'azione per leggere la pagina strappata, per cui è impossibile ottenere le capacità *premere* (indispensabile per continuare) e *piegarsi*: il gioco è, di fatto, perso in partenza.

Si può quindi offrire un'introduzione alle capacità fin dall'inizio: si dovrà aggiungere un oggetto (un poster) situato nella cabina. Se guardato o letto, esso spiegherà al giocatore la nuova feature e permetterà di apprendere premere/piegarsi.

È stato scelto un poster in cabina al posto della pagina strappata per due motivi:

- L'idea di base è di recarsi nel luogo in cui c'è la pagina e selezionare l'abilità da cambiare, come accade in alcuni videogiochi in cui bisogna ritornare in un posto con ricorrenza per andare avanti nella storia (come i falò nei cosiddetti souls-like games). Tuttavia, una pagina strappata è trasportabile, quindi sarebbe possibile cambiare abilità ovunque. È meglio sostituirla con un oggetto leggibile (appunto il poster) ma non prendibile
- La cabina è l'ambiente iniziale, per cui è il posto giusto per l'introduzione al giocatore delle capacità

Il collega Vergine non ha introdotto un metodo per ripristinare una delle capacità base nel caso in cui essa sia stata cancellata, per cui si può pensare a un personaggio che ti può ricompensare se esegui una certa azione: il vagabondo, a cui dare una mela per ricordare una capacità base a scelta. Quest'azione è eseguibile solo una volta, altrimenti se ne abuserebbe. L'azione *elimina capacita'* potrebbe essere migliorata mostrando le capacità in possesso del giocatore, prima di chiedere quale eliminare.

3.28.1 Struttura dati

Ogni capacità è un oggetto della classe *Capacita_singola*, e la gestione delle capacità avviene mediante la classe *Capacita* che richiede anche l'utilizzo della lista

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
- gettitolo() → 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 avvertirà 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'identita'" 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 contraddistinguere 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 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: 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, sediti “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: NNSSEOOOUUDD, 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

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	Restuisci 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:
NNSSSEOOUUDD, 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:

NNSSSEOOUUDD

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
	batterie	16	-41
una chiave_inglese	chiave_inglese	118	41
un diario	diario	57	-41
un cartello	cartello	60	-41
un computer	computer	99	-41
un meccanico	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;

VVVV 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”, LUOGO “CASERMA” E OGGETTO CASSAFORTE

4.16.1 Verbi “ruba” e “rubati” e azione “apri cassaforte”

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

Mentre per la creazione dei comandi per interagire con la cassaforte e la cella verrà inserito nel gioco il vocabolo il vocabolo “cassaforte” e “cella”.

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
cassaforte	203
cella	201

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
Guarda cella	4600100201	153
Guarda vetrina	4600100019	154
Guarda cassaforte	4600100203	155
Apri cassaforte	4600220203	156

Nel gioco sono stati aggiunti gli oggetti nella caserma per poterci interagire, nel gioco gli oggetti sono rappresentati come segue:

- *Nome oggetto:* identifica il nome dell'oggetto;
- *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.

Nome oggetto	Codice OO	Codice mappa
Una cassaforte	123	-46
Una vetrina	122	-46
Una cella	121	-46

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 essere raggiunto tramite altre stanze e quando il "carabiniere" arresta 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 NNSSEEOOUUDD
(N = Nord, S = Sud, E = est, O = ovest, U = Salì, D = Scendi);
- **Codice Luogo:** codice numerico univoco che lo contraddistingue dagli altri luoghi presenti nel gioco.

ETICHETTA	CODICE MAPPA	CODICE LUOGO
Caserma	00 00 00 03 00 00	53

4.17 CHIESA

4.17.1 Aggiunta oggetti e vocaboli nel gioco

In Astro.cpp sono stati aggiunti i nuovi Oggetti previsti in fase di progettazione, ovvero la "Croce", la "Panchina", il "Confessionale", il "Sacerdote" e i rispettivi Vocaboli, ordinati in ordine crescente di codice. Inoltre, sono anche stati aggiunti i nuovi vocaboli "Siediti", "Prega", "Confessati", e le seguenti azioni:

- "confessati" e "entra confessionale" che sono sinonimi;
- "siediti" e "siediti panchina" che sono sinonimi;
- "parla sacerdote";
- "prega".

OGGETTI:

```
oggetti.inserisci(Oggetto("unacroce",43 , 20));
oggetti.inserisci(Oggetto("unapanchina",92 , -20));
oggetti.inserisci(Oggetto("unconfessionale",97 , -20));
oggetti.inserisci(Oggetto("unsacerdote",98 , -20));
```

VOCABOLI:

```
vocabolario.inserisci("croce",43);
vocabolario.inserisci("panchina", 92);
vocabolario.inserisci("siediti", 94);
```

```

vocabolario.inserisci("prega", 95);
vocabolario.inserisci("confessati", 96);
vocabolario.inserisci("confessionale", 97);
vocabolario.inserisci("sacerdote", 98);

```

AZIONI:

```

azioni.inserisci(209600, 70); // Azione per confessarticoncomandoconfessati
azioni.inserisci(207497, 70); //Azione per
confessarticoncomandoentraconfessionale
azioni.inserisci(209400, 71); // Azione per sederticoncomandosiediti
azioni.inserisci(209492, 71); // Azione per sederticoncomandisieditipanchina
azioni.inserisci(200200, 72); // Azionedisupportoalla 71
azioni.inserisci(203098, 73); // Azione per parlarealsacerdote
azioni.inserisci(209500, 74); // Azione per pregare

```

4.18 UFFICIO / DEPOSITO

4.18.1 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
62	Documento	Documento	-12

4.18.2 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
80	Consegna	Consegna documento
42	Ritira	Ritira documento

4.18.3 Comandi

- **consegna documento** che permette di consegnare il documento nell'inventario del giocatore;

- **ritira documento** che permette di ritirare dall’Ufficio / deposito uno dei documenti disponibili.

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
140	1200800062	Consegna documento
141	1200420062	Ritira documento

4.19 SALI & TABACCHI

Per rappresentare i luoghi del gioco bisogna assegnare delle caratteristiche univoche, in modo da facilitarne l’elaborazione, la lettura e la comprensione. questo si stila una tabella con le seguenti caratteristiche:

- **Etichetta**: stringa che identifica il nome del luogo in modo esaustivo;
- **Codice Luogo**: stringa numerica che identifica i percorsi utilizzabili all’interno del luogo, composta da 12 caratteri *NNSSEEOOUUDD* aventi il seguente significato:
(N=Nord, S=Sud, E=Est, O=Ovest, U=Sali, D=Scendi)
- **Codice Mappa**: stringa numerica univoca utile per differenziarla dagli altri luoghi presenti nel gioco.

ETICHETTA	CODICE LUOGO	CODICE MAPPA
Sali&Tabacchi	000000250000	43

È possibile accedere a “Sali&Tabacchi” esclusivamente dall’aula (con codice 25), perciò per raggiungere il nuovo luogo partendo dal luogo iniziale (“La tua cabina”) bisognerà andare ad est di esso, arrivando nel corridoio; poi andando a sud si arriva all’estremità sud del corridoio. Una volta arrivati a questo punto basterà andare nuovamente a sud e poi ad ovest per arrivare nel Sali&Tabacchi.

4.19.1 Oggetti

Per ciascun oggetto sono state assegnate delle caratteristiche univoche:

- **Etichetta**: stringa che indica il nome dell’oggetto;
- **Vocabolo**: rappresenta la parola chiave riconosciuta dal programma per poter effettuare l’azione che include un determinato oggetto.
- **Codice oggetto**: rappresenta il codice identificativo di un oggetto del gioco.
- **Codice mappa**: rappresenta la stanza in cui è presente l’oggetto; il simbolo meno indica che l’oggetto non è trasportabile;

ETICHETTA	VOCABOLO	CODICE LUOGO	CODICE MAPPA
Sale	Sale	102	43
Sigarette	Sigarette	101	43

Lotteria	Lotteria	105	43
Orari	Orari	103	43
Elaboratore Bonus	Elaboratore bonus	104	43

4.19.2 Azioni

Per poter interagire con ogni oggetto oltre ad assegnargli delle caratteristiche univoche, è necessario assegnargli le azioni che si intende eseguire con esso:

Azione: indica l'azione che si vuole eseguire con l'oggetto.

Vocabolo: indica la parola chiave riconosciuta dal sistema per svolgere una determinata azione.

Codice oggetto: indica il codice univoco con cui è rappresentata l'azione all'interno del gioco.

AZIONE	VOCABOLO	CODICE OGGETTO
Comprare	Compra	33
Acquistare	Acquista	33
Giocare	Gioca	100
Leggere	Leggi	25
Guardare	Guarda	10
Vedere	Vedi	106
Prendere	Prendi	8
Ottenerе	Ottieni	107
Usare	Usa	29

4.19.3 Comandi

Per ogni comando utilizzabile all'interno del gioco sono state assegnate delle caratteristiche univoche:

- **Comando:** indica ciò che eseguirà il giocatore.

- **Codice azione:** indica il codice identificativo per poter riconoscere il comando all'interno del gioco.

- **Codifica**, ovvero il codice numerico ottenuto tramite l'espressione:

$$LL * 100000000 + VVVV * 10000 + OOOO$$

Dove:

- LL = Codice del luogo in cui il comando può essere impartito;

- VVVV = Codice del verbo che indica una certa azione;

- OOOO = Codice dell'oggetto.

COMANDO	CODIFICA	CODICE AZIONE
Compra sigarette	4300330101	144
Acquista sigarette	4300330101	144
Prendi sigarette	4300080101	144
Compra sale	4300330102	143

Acquista sale	4300330102	143
Prendi sale	4300080102	143
Gioca lotteria	4301000105	142
Guarda orari	4300100103	140
Leggi orari	4300250103	140
Vedi orari	4301060103	140
Usa elaboratore	4300290104	145
Usa elaboratore bonus	4300290104	145
Ottieni bonus	4301070104	145

Nel progetto da integrare era presente un bug sul comando “guarda orari”, in quanto una volta digitato dal giocatore gli orari non venivano mostrati ed usciva la scritto “Non noto nulla di particolare”. Ho risolto tale problema aggiustando semplicemente la codifica del comando.

Da 3701000103 a 4300100103

Si può notare che il bug era dovuto all’errato inserimento del codice del verbo “guarda”.

4.20 DISTANZA/SFORZO

I file estratti dal progetto di De Rinaldis, aggiunti poi a tale progetto (riportandoli senza alcuna modifica), con il fine di realizzare correttamente la funzionalità di calcolo della distanza/sforzo sono: “Grafo.h” (per la struttura dati del grafo), “Nodo_Grafo.h” e “Nodo_Grafo.cpp” (per il nodo del grafo), “Contapassi.h” e “Contapassi.cpp” (per integrare la funzione già implementata).

Tali modifiche e integrazioni sono state necessarie per non alterare il corretto funzionamento della maggior parte delle componenti della funzionalità; si è deciso, dunque, di includere tali classi e file adattandoli al progetto in modo tale da renderli compatibili con le altre funzionalità già presenti.

Per quanto riguarda il termine “sforzo” (che avvia la funzione vera e propria), esso si inserisce sia all’interno del vocabolario che all’interno dell’insieme delle azioni.

Prendendo in esame il termine all’interno del vocabolario, esso avrà la seguente struttura:

ETICHETTA	CODICE
sforzo	99

Al termine “sforzo” nel dizionario è associato un codice univoco di valore 99: tale termine verrà associato all’azione di visualizzazione delle informazioni sulla distanza e sullo sforzo fisico che l’utente potrà eseguire richiamando la stessa parola.

Definendo l’azione associata al verbo “sforzo”, si ha la seguente struttura:

VERBO	CODIFICA	CODICE
sforzo	990000	147

Analizzando nel dettaglio tale record:

- Verbo: termine con il quale si può invocare l'azione di visualizzazione delle informazioni distanza/sforzo
- Codifica: valore numerico per i vari parametri dell'azione.

Nel particolare, esso segue la sintassi:

- LLLL: Codice del luogo in cui il comando può essere eseguito (0000, ovvero tutte le zone della mappa)
- VVVV: Codice del verbo/termine associato all'azione (99, ovvero il termine "sforzo")
- OOOO: Codice dell'oggetto sul quale si può eseguire l'azione (0000, ovvero nessun oggetto)

Codice: valore univoco associato all'azione da compiere

4.21 FABBRICA

Per l'aggiunta del luogo “*Fabbrica*” sono stati modificati i seguenti file:

- *Mappa.nav*
- *Astro.h*
- *Astro.cpp*
- *Portafoglio.cpp* (risoluzione di un piccolo bug per usufruire della funzione *hai_Portafoglio*)

Sono stati aggiunti inoltre i seguenti file:

- *fabbrica.h*
- *fabbrica.cpp*

4.21.1 Inserimento luogo nella mappa

Il luogo “*Fabbrica*” si troverà ad est della Sala Giochi. Per raggiungerlo, partendo dalla cabina bisogna spostarsi ad est (Corridoio), nord (estremità Nord del corridoio), scendi (Ufficio/Deposito), est (Negozio), est (Banca), est (Sala Giochi), est (*Fabbrica*).

4.21.1.1 Modifica Mappa.nav

È stato necessario modificare il file *Mappa.nav* per inserire il luogo “*Fabbrica*” aggiungendo il luogo n°44 e modificando il luogo “*Sala Giochi*” (n°35) per poter raggiungere tramite il comando est la *Fabbrica*.

I luoghi sono definiti da tre componenti:

- *Codice Luogo*: identificativo intero che distingue i luoghi all'interno della mappa;
- *Nome Luogo*: stringa che descrive il luogo;
- *Codice direzioni*: stringa numerica che indica i luoghi confinanti con esso: NNSSEEOOUUDD (Nord, Sud, Est, Ovest, Up(sali), Down(Scendi)).

Codice Luogo	Nome Luogo	Codice direzioni
44	Nella fabbrica	000000350000
35	Nella sala giochi	0000 44 150000

4.21.2 Oggetti

È stato inserito l'oggetto "fabbricante" che si troverà nella fabbrica (luogo n°44), e non sarà trasportabile. Il suo codice identificativo è 108.

Gli oggetti sono definiti da quattro componenti:

- Etichette: stringa che contiene il nome dell'oggetto;
- Vocabolo: stringa per interagire con l'oggetto;
- Codice Oggetto: intero identificativo dell'oggetto;
- Codice Mappa: luogo in cui l'oggetto è posizionato
(0 → inventario, <0 → non trasportabile, >0 → trasportabile).

Etichetta	Vocabolo	Codice Oggetto	Codice Mappa
un fabbricante	fabbricante	108	-44

4.21.3 Azioni

È stata inserita l'azione n° 148, che permetterà al protagonista di parlare con il fabbricante tramite il comando usando il verbo "parla".

Le azioni sono definite da tre componenti:

- Codifica: stringa numerica che indica i vocaboli che richiamano l'azione, nel formato: LLVVVVOOOO (L= luogo in cui può avvenire, V= codice del verbo, O = codice dell'oggetto);
- Azione: intero che identifica quale azione eseguire.

Codifica	Azione
4400300108	148

44 → fabbrica;
0030 → verbo parla;
0108 → oggetto fabbricante

4.22 TELETRASPORTA

4.22.1 Verbo "teletrasporta"

L'integrazione della funzionalità richiede la creazione del verbo "teletrasporta" e quindi il suo inserimento all'interno del vocabolario del gioco.

Esso è formato da:

- Etichetta: che identifica il nome del verbo;
- Codice del luogo: codice univoco del verbo

Etichetta	Codice
Teletrasporta	135

4.22.2 Azione teletrasporta

Al verbo “teletrasporta” viene assegnata un’azione che sarà richiamata al momento del suo utilizzo.

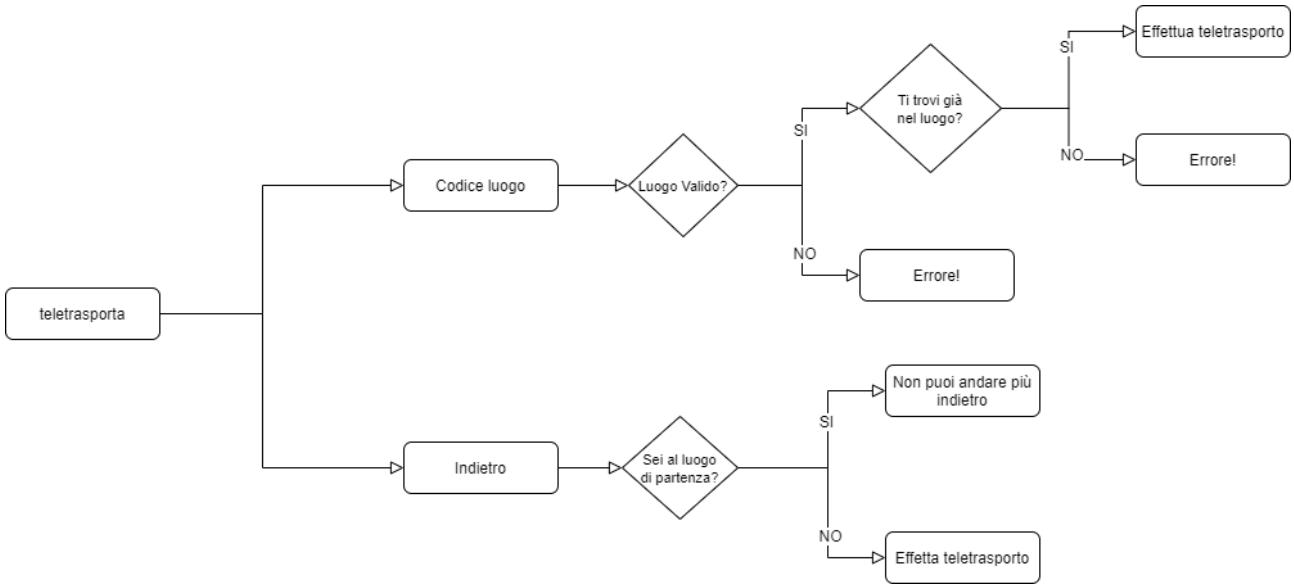
Essa è composta da:

- Codifica: rappresenta il codice univoco che identifica il comando. Viene ottenuta tramite l’espressione LLLLVVVVOOOO, dove “LLLL” indica il luogo in cui si può compiere il comando (0000 nel caso in cui l’azione può essere usata in qualunque stanza), “VVVV” indica il codice del verbo con cui è associato e “OOOO” l’oggetto su cui si compie il comando (0000 se l’azione non può essere applicata su nessuno oggetto).
- Codice: codice identificativo dell’azione, se esso è positivo l’azione potrà essere applicata su qualunque luogo presente nella mappa del gioco.
- Verbo: verbo associato all’azione.

Codice	Codifica	Verbo
149	135000	Teletrasporta

Il giocatore quindi potrà richiedere in qualunque istante durante il gioco, indipendentemente da dove si trova, di utilizzare la funzione di teletrasporto.

Una volta utilizzato il comando il giocatore potrà scegliere se andare in un luogo da lui indicato o se tornare indietro, seguendo uno degli scenari rappresentati:



4.23 CASSAFORTE E CASSETTA DI SICUREZZA MALMESSA

4.23.1 Luogo "Cassaforte"

Nel gioco, ogni luogo è identificato da tre attributi:

- **Etichetta:** stringa che identifica il nome del luogo;
 - **Codice mappa:** 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).
- Questi caratteri verranno poi sostituiti dal Codice luogo del luogo al quale si vuole accedere prendendo quella specifica strada, 00 se non si può andare.
- **Codice luogo:** codice numerico a due cifre univoco che distingue ogni luogo da un altro.

Il luogo quindi è implementato così:

Etichetta	Codice Mappa	Codice Luogo
Cassaforte	150000000000	54

Nella mappa standard vi è uno sfalsamento a partire dal quarantaquattresimo luogo perché da quell'id in poi, questi smettono di seguire il normale ordine visto che i luoghi con gli id ordinati sono su altre mappe, difatti gli id sono: 44,48,53,54. Per rendere funzionale questo sistema, bisogna considerare l'ordine effettivo 44,45,46,47 e quindi, per il luogo 48 considerare uno sfalsamento di 3

(48-3=45) e per i luoghi 53 e 54 uno sfalsamento di 7 (rispettivamente, 53-7=46 e 54-7=47). Per cui, per riferirsi alle azioni e agli oggetti nella Cassaforte bisogna riferirsi al luogo 47.

4.23.2 Oggetto “Cassetta di sicurezza malmessa”

Nel gioco, ogni oggetto è identificato da tre attributi:

- **Codice oggetto:** codice univoco a tre cifre che identifica l'oggetto;
- **Nome oggetto:** stringa contenente il nome dell'oggetto;
- **Codice luogo:** codice del luogo in cui si trova l'oggetto. Preceduto da un – se questo oggetto non è trasportabile.

L'oggetto quindi è implementato così:

	Nome oggetto	Codice Luogo
120	Una cassetta di sicurezza malmessa	-47

4.23.3 Azioni “Deposita oggetto”, “Controlla cassetta”, “Ritira oggetto”

Nel gioco, ogni azione è identificata da tre attributi:

- **Comando:** stringa composta da una o più parole necessarie per eseguire una determinata azione o scaturite da un determinato evento;
- **Codifica:** codice numero ottenuto dall'espressione

$$[\text{Codice}_\text{Luogo} * 1000000 + \text{Codice}_\text{Verbo} * 10000 + \text{Codice}_\text{Oggetto}]$$
che permette di avere una codifica univoca;
- **Codice azione:** codice identificativo dell'azione.

Le azioni implementati da Carlucci sono identificate da:

Comando	Codifica	Codice Azione
Deposita oggetto	4700099999	150
Controlla cassetta	4700350041	151
Ritira oggetto	4700429999	152

Da notare che i verbi utilizzati (“deposita” 9, “controlla” 35, “ritira” 42) erano già presenti nel gioco, così come il vocabolo “cassetta”, 41, e il codice 9999 indica che qualsiasi oggetto può essere depositato/ritirato.

4.24 ERROR! REFERENCE SOURCE NOT FOUND.

L'integrazione delle funzionalità richiede la creazione dei verbi “ritorna” e “resoconto” e quindi il loro inserimento all'interno del vocabolario del gioco.

Essi sono formati da:

- Etichetta: che identifica il nome del verbo;
- Codice del luogo: codice univoco del verbo

Etichetta	Codice
Ritorna	72
Resoconto	12

4.24.1 Azione ritorna

Al verbo “ritorna” viene assegnata un’azione che sarà richiamata al momento del suo utilizzo.

Essa è composta da:

- Codifica: rappresenta il codice univoco che identifica il comando. Viene ottenuta tramite l’espressione LLLLVVVVOOOO, dove “LLLL” indica il luogo in cui si può compiere il comando (0000 nel caso in cui l’azione può essere usata in qualunque stanza), “VVVV” indica il codice del verbo con cui è associato e “OOOO” l’oggetto su cui si compie il comando (0000 se l’azione non può essere applicata su nessuno oggetto).
- Codice: codice identificativo dell’azione, se esso è positivo l’azione potrà essere applicata su qualunque luogo presente nella mappa del gioco.
- Verbo: verbo associato all’azione.

Verbo	Codifica	Codice
Ritorna	720000	72

Il giocatore quindi potrà richiedere in qualunque istante durante il gioco, indipendentemente da dove si trova, di utilizzare la funzione ritorna.

4.24.2 Azione resoconto

Al verbo “resoconto” viene assegnata un’azione che sarà richiamata al momento del suo utilizzo.

Essa è composta da:

- Codifica: rappresenta il codice univoco che identifica il comando. Viene ottenuta tramite l’espressione LLLLVVVVOOOO, dove “LLLL” indica il luogo in cui si può compiere il comando (0000 nel caso in cui l’azione può essere usata in qualunque stanza), “VVVV” indica il codice del verbo con cui è associato e “OOOO” l’oggetto su cui si compie il comando (0000 se l’azione non può essere applicata su nessuno oggetto).
- Codice: codice identificativo dell’azione, se esso è positivo l’azione potrà essere applicata su qualunque luogo presente nella mappa del gioco.
- Verbo: verbo associato all’azione.

Verbo	Codifica	Codice
Resoconto	120000	12

Il giocatore quindi potrà richiedere in qualunque istante durante il gioco, indipendentemente da dove si trova, di utilizzare la funzione di resoconto.

4.25 LABORATORIO ANALISI

4.25.1 Luogo

Il luogo "Laboratorio analisi" è raggiungibile proseguendo verso est da "La tua cabina", salendo si arriva al pronto soccorso e poi ad est ci sarà il luogo aggiunto.

I luoghi vengono definiti tramite i seguenti parametri:

- Codice luogo: Codice numerico univoco che identifica il luogo
- Etichetta: Stringa contenente il nome del luogo
- Codice delle direzioni: Stringa che consente di sapere le direzioni in cui è possibile andare (partendo dal luogo Laboratorio analisi). È composta da sei sottostringhe contenenti ciascuna due caratteri che identificano rispettivamente le seguenti direzioni: NNSSEEOOUUDD (N = nord, S = sud, E = est, O = ovest, U = Salì, D = Scendi). Sapendo che il luogo "Laboratorio analisi" è posizionato a est del "Pronto Soccorso" e che il codice identificativo del luogo "Pronto soccorso" è 19, si ha:

CODICE LUOGO	ETICHETTA	CODICE
55	Laboratorio analisi	000000190000

4.25.2 Oggetti

Gli oggetti vengono definiti tramite i seguenti parametri:

- Codice oggetto: Codice numerico univoco che identifica quel particolare oggetto all'interno del gioco
- Etichetta: Stringa che identifica il nome dell'oggetto
- Vocabolo: Parola chiave che il giocatore deve utilizzare per interagire con quel particolare oggetto.
- Codice luogo: Codice del luogo in cui si trova l'oggetto.

Il segno "--" indica che l'oggetto non è trasportabile.

CODICE OGGETTO	ETICHETTA	VOCABOLO	CODICE MAPPA
109	Un droide medico	utilizza droide	-48

4.25.3 Azioni

Per l'implementazione è stata inserita l'azione: "Utilizza droide"

Le azioni vengono definite tramite i seguenti parametri:

- Comando: Stringa composta da una o più parole necessarie per eseguire una determinata azione o un determinato evento.
- Codifica: Codice numerico, nel progetto base viene ottenuto dall'espressione matematica:
[CodiceLuogo * 100000000 + CodiceVocabolo *10000 + CodiceOggetto]
- Codice azione: Codice numerico univoco che identifica l'azione.

I codici vocabolo di "droide" e "utilizza" implementati all'interno del gioco corrispondono rispettivamente a 201, 202.

COMANDI	CODIFICA	CODICE AZIONE
"Utilizza droide"	$48 * 100000000 + 202 * 10000 + 201 = 4702020201$	157

4.26 GUARDA MAPPA

4.26.1 Verbo “guarda mappa”

L'integrazione della funzionalità richiede la creazione del verbo “guarda mappa” e quindi il suo inserimento all'interno del vocabolario del gioco.

Esso è formato da:

- Etichetta: che identifica il nome del verbo;
- Codice del luogo: codice univoco del verbo
-

Etichetta	Codice
Guarda mappa	158

4.26.2 Azione “guarda mappa”

Al verbo “guarda mappa” viene assegnata un'azione che sarà richiamata al momento del suo utilizzo.

Essa è composta da:

- Codifica: rappresenta il codice univoco che identifica il comando. Viene ottenuta tramite l'espressione LLLLVVVVOOOO, dove “LLLL” indica il luogo in cui si può compiere il comando (0000 nel caso in cui l'azione può essere usata in qualunque stanza), “VVVV” indica il codice del verbo con cui è associato e “OOOO” l'oggetto su cui si compie il comando (0000 se l'azione non può essere applicata su nessuno oggetto).
- Codice: codice identificativo dell'azione, se esso è positivo l'azione potrà essere applicata su qualunque luogo presente nella mappa del gioco.
- Verbo: verbo associato all'azione.

Codice	Codifica	Verbo
158	100019	Guarda mappa

4.27 CUCINA

è stato integrato il nuovo luogo "Cucina", per permettere al Giocatore di svolgere delle nuove attività. Il luogo è identificato da:

- Un codice identificativo univoco di tipo numerico che lo distingue dagli altri luoghi presenti nel Gioco
- Un codice di tipo numerico che identifica i percorsi utilizzabili all'interno del luogo. Tale codice è formato da dodici cifre aventi il seguente significato: NNSSEEOOUUDD dove N=Nord, S=Sud, E=Est, O=Ovest, U=Sali, D=Scendi;
- Un'etichetta di tipo testuale che identifica il nome del luogo

CODICE LUOGO	ETICHETTA	CODICE DIREZIONE
56	Nella cucina	000600000000

4.27.1 OGGETTI

Gli oggetti vengono definiti tramite i seguenti parametri:

- Codice oggetto: Codice numerico univoco che identifica quel particolare oggetto all'interno del gioco
- Etichetta: Stringa che identifica il nome dell'oggetto
- Vocabolo: Parola chiave che il giocatore deve utilizzare per interagire con quel particolare oggetto.
- Codice luogo: Codice del luogo in cui si trova l'oggetto. (Il segno "-" indica che l'oggetto non è trasportabile.)

CODICE OGGETTO	ETICHETTA	VOCABOLO	CODICE MAPPA
677	mela	mela	49
678	barretta_energetica	barretta_energetica	49
46	pizza	pizza	49
680	bimby	bimby	-49
681	tavolo	tavolo	-49
682	frigorifero	frigorifero	-49
683	televisione	televisione	-49

4.27.2 AZIONI

Le azioni vengono definite tramite i seguenti parametri:

- Comando: Stringa composta da una o più parole necessarie per eseguire una determinata azione o un determinato evento.
- Codifica: Codice numerico
- Codice azione: Codice numerico univoco che identifica l'azione.

Comandi	Codifica	Codice azione
mangia barretta	4900980678	159
mangia pizza	4900980046	160
apri frigorifero	4900220682	161
guarda televisione	4900100683	162
preparazione cibo	4900800680	163
mangia mela	4900980677	164

4.28 CAPACITÀ

In fase di realizzazione sono state opportunatamente modificate le azioni che richiedono capacità, in modo tale da stampare un messaggio di errore nel caso in cui quella richiesta sia assente.

In più sono state le modifiche descritte in fase di progettazione.

Le capacità sono:

CAPACITÀ	CODICE VOCABOLO	CODICE MAPPA
Aprire	22	0
Digitare	92	0
Leggere	25	0
Scrivere	19	0
Piegarsi	58	-99
Premere	26	-99
Saltare	49	-99

Guidare	48	-99
---------	----	-----

Si è ritenuto non necessario l'inserimento di nuove capacità.

4.28.1 MAPPATURA

Non è stato aggiunto né rimosso alcun luogo, perché le capacità riguardano il personaggio.

4.28.2 VOCABOLI

Tutti i vocaboli inseriti dal collega Vergine Erik si trovano anche nel progetto base.

I vocaboli inseriti/modificati* sono:

VOCABOLO	CODICE
foglio*	72
poster	166
vagabondo	167

Foglio è presente nel progetto base, ma è commentato: il collega Vestita Salvatore lo ha sostituito con *siediti*. Poiché la traccia richiedeva di implementare tutte le funzionalità introdotte da Vergine, tra cui il gioco dell'impiccato tramite la lettura del foglio, esso è stato aggiunto di nuovo. È stato scelto il codice 72 perché inutilizzato.

4.28.3 OGGETTI

Gli oggetti inseriti sono:

ETICHETTA	VOCABOLO	CODICE OGGETTO	CODICE MAPPA
un foglio	foglio	72	-6
un poster	poster	166	-6
un vagabondo	vagabondo	167	-4

Nonostante un foglio imprendibile (come era stato implementato da Vergine) non abbia molto senso, è stato lasciato così perché altrimenti sarebbe possibile giocare all'impiccato in qualsiasi momento e si avrebbe quindi troppo vantaggio.

4.28.4 COMANDI

I comandi aggiunti/modificati* sono:

COMANDO	CODIFICA	CODICE AZIONE
Mangia mela*	980677	164
Elenca capacita	430026	166
Elimina capacita	180026	167
Guarda motorino	1200100083	168
Leggi foglio	600250072	169
Guarda poster	600100166	170
Leggi poster	600250166	170
Parla vagabondo	400300167	171

Mangia mela, inserito da Costantini Andrea, è ora possibile in tutti i luoghi, se è stata presa.

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 più completi, questo animale ha più 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 *lathos* (pietra) e indica un escremento, prodotto da un animale vissuto nel passato, che si è 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 è 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 è 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 è 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 *Pterodactylus* erano 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 più diffuso geograficamente degli altri tirannosauridi. I suoi fossili si trovano in una varietà 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 grigliate, 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 qualita' di ingredienti, modalita' 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 "signora della casa" 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 all'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 semplicita' 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 divinita' funerarie. Nei due registri centrali, meglio conservati, Horemheb e' colto in momenti diversi: seduto in compagnia della sua "animaa" 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 dell'oltretomba per garantirsi la sopravvivenza eterna. Il serpente ureo, simbolo di regalita', che orna la sua fronte, fu scalpellato in un secondo momento, solo quando Horemheb divenne faraone dell'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 attivita' 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 nell'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 già spennati, contenitori per liquidi, etc.), perché il defunto possa nutrirsi per forza magica nella vita ultraterrena. Sullo stesso lato, a una delle estremità, è 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 sepoltura. ";

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 succlistista(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.finlista(ind2))
    {
        inslista(b.leggilista(ind2),ind);
        ind = succlist(a(ind));
        ind2 = b.succlist(a(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>::preclista(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());
        suclista(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(){}

```

```

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 grafovusto() 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))// 1-----+
    {
        //|

```

```

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
nelementi--; //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.finelista(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.leggilista(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.finellista(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;

//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,tipoPeso);
    ~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,tipoPeso>::Adiacenza() //costruttore generico
{

```

```
}
```

```
template <class nodo,class tipoPeso> Adiacenza<nodo,tipoPeso>::Adiacenza(nodo n, tipoPeso 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);
```

```

}

//sovraffaccarico 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=="s¬" || 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.finelista(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.leggilista(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 variabile 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.....PRESENT | ");
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_indotinata;
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: ");
    }
}
while(indovina == false);

```

```

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;
}

}

```

```

{
    interfaccia.scrivi("Ok ci vediamo");
}

}

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

//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(" | Sistema Archivio 2.0 | ");
    interfaccia.scrivi(" | Benvenuto Comandante nel sistema di archivio |
dell'astronave. | ");
    interfaccia.scrivi(" | A causa dei gravi danni subiti, l'integrita' dell'archivio e' |
compromessa. | ");
    interfaccia.scrivi(" | Il sistema e' stato in grado di ripristinare solo alcuni file. |
");
    interfaccia.scrivi(" | Inoltre vi e' un problema nella visualizzazione a schermo dei |
file. | ");
    interfaccia.scrivi(" | I file possono pero' essere stampati senza problemi |
");
    interfaccia.scrivi(" | Per ritirare il documento digitare il numero corrispondente. |
");
    interfaccia.scrivi(" | Attenzione: problemi tecnici rallenteranno il processo di |
stampa. | ");
    interfaccia.scrivi(" | Per uscire digitare il numero corrispondente. |
");
    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");
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(){

    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");
    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":

- bigietto_museo
- mostra_galleria_a
- mostra_galleria_b
- mostra_galleria_c
- mostra_galleria_d

La funzione *bigietto_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::bigietto_museo() //funzione ausiliare per controllare la disponibilita' del  
portafoglio  
// e acquistare bigietto 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. | ");

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 lithos (pietra) e | ");

interfaccia.scrivi(" | indica un escremento, prodotto da un animale vissuto nel passato, che si è | ");

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'è™ escremento e' stato prodotto da un erbivoro, mentre se si individuano | ");
```

```
interfaccia.scrivi(" | frammenti di ossa, artigli o tendini l'è™ 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(" |-----|");  
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 scorcio
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 . | ");
tempo--;
interfaccia.scrivi(" |-----| ");
system("pause");
break;
case 3:
system("cls");
interfaccia.scrivi(" |-----| ");
interfaccia.scrivi(" |Combattimento degli Orazi e Curiazi(1612-1613) | ");
interfaccia.scrivi(" |-----| ");
interfaccia.scrivi(" |Episodio della guerra di Roma contro la vicina città di Albalonga che si | ");
interfaccia.scrivi(" |concluse con un duello tra i rappresentanti di Roma, gli Orazi, e quelli di | ");
interfaccia.scrivi(" |Albalonga, i Curiazi. Gli eserciti contendenti assistono alla scena finale | ");
interfaccia.scrivi(" |del duello, quando l' ultimo degli Orazi sta per colpire l' ultimo degli | ");
interfaccia.scrivi(" |avversari | ");
interfaccia.scrivi(" |-----| ");

```

```

tempo--;
system("pause");
break;
case 4:
system("cls");
interfaccia.scrivi(" | -----| ");
interfaccia.scrivi(" | Ratto delle Sabine (1635-1636) | ");
interfaccia.scrivi(" | -----| ");
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");

}

}

while(uscita_museo!=true);

}

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(" |Sono stati ritrovati numerosi reperti risalenti a questa epoca, alcuni dei quali |");
interfaccia.scrivi(" |sono presenti all'interno della nostra galleria. |");
interfaccia.scrivi(" |Per maggiori descrizioni, consultare la guida elettronica: |");
interfaccia.scrivi(" |1 - Stele funeraria del tipo a nome di Sameri |");
interfaccia.scrivi(" |2 - Rilievo con scena di libagione |");
interfaccia.scrivi(" |3 - Rilievo dalla tomba di Horemheb con scena di lavoro nei campi dell'oltretomba |");
interfaccia.scrivi(" |4 - Rilievo con la dea Renenutet |");
interfaccia.scrivi(" |5 - Sarcofago a cassa a nome di Irinimenpu |");
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(" |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(" |l'iscrizione in caratteri geroglifici sottostante completa per ");
        forza magica | );

```

```
interfaccia.scrivi(" | L'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 sembianze 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 attivitÃ agricole: nella parte alta, alcuni
contadini | ");

interfaccia.scrivi(" |puliscono il grano lanciandolo in aria con sessole di legno per
separare i | ");

interfaccia.scrivi(" |chicchi dalla 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
nellâ€™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
funeraria | ");

    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(" |nutrirsene 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 caropforo  
(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:

system("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.legglista(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.legglista(p)==2){
```

```
        interfaccia.scrivi("È UN LIBRO DI UNGARETTI");
```

```
        interfaccia.scrivi("L'hai già 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.finelista(p)!=true){//fino a quando non raggiungiamo finelista
        if(l.leggilista(p)==3){

            interfaccia.scrivi("È UN LIBRO DI LEOPARDI");
            interfaccia.scrivi("L'hai già 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.");

}

story_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>>sceltainvia;
    switch(sceltainvia)
    {
        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 gia' 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 gia' ritirato la lettera +");
}

else
{
    system("cls");
    interfaccia.scrivi(" \n+-----+");
    interfaccia.scrivi(" + Attenzione! +");
    interfaccia.scrivi(" + Recupera prima il tuo documento d'identita' +");
    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 leggitstato(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             indice++;
2514         }
2515         indice++;
2516         int i=0;
2517         while(rigafile[indice]!='*'){
2518             statodialogostr[i]=rigafile[indice]; // leggo lo stato del dialogo
2519             indice++;
2520             i++;
2521         }
2522         statodialogo=atoi(statodialogostr);

```

```

2523     indice++;
2524     while(rigafile[indice]!='#'){
2525         if(rigafile[indice]=='$')
2526             frasedialogo=frasedialogo+'\n';
2527         else
2528             frasedialogo=frasedialogo+rigafile[indice]; // leggo la frase da inserire nel nodo Domanda
2529         indice++;
2530     }
2531     if (nome==nomepersonaggio)
2532         dialogo-
2533     >inseriscidomanda(statodialogo,frasedialogo); //inserisce la Domanda nel Dialogo

```

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
2710             el file
2711         }
2712     }
2713     filestati.close();
2714     return stato;

```

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]!='|'){
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 scrivistatosufile:

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::scrivistatosufile(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
1364     ifstream file(fStringa.c_str(), ios::in);
1365 //modifiche effettuate da D'Andria Dresden sul controllo del caricamento da file
1366     ifstream astrostatidialoghi("Astrostatidialoghi.txt");
1367     bool astrostatidialoghiaperto=astrostatidialoghi.good();
1368     bool fileaperto=file.good();
1369     if(fileaperto || astrostatidialoghiaperto){
1370         interfaccia.scrivi("Ripristino partita...");
1371         if(fileaperto){
1372             for (i = 1; i <= oggetti.get_n_oggetti(); i++){
```

```

1373         file >> valore;
1374         oggetti.set_luogo(i,valore);
1375     }
1376     file >> luogo_attuale;
1377     file >> tempo;
1378     file >> passo_soluzione;
1379     load_specifiche(file);
1380     bacheca.CaricaBacheca(file);
1381     file.close();
1382 }
1383 if(astrostatidialoghiaperto)
1384     caricaStatiDialoghi(); //modifiche D'Andria Dresda
1385 ifstream jukeLuci("JukeLuci.txt");
1386 if (jukeLuci.good() )
1387     caricaJukeBoxLuci();
1388 }
1389 else
1390     interfaccia.scrivi("Non ci sono partite salvate!");
1391 //INIZIO MODIFICHE CICALA GIACOMO
1392 file >> numEuro;
1393 for(int j=1; j<=numEuro; j++){
1394     file >> slot;
1395     slotmachine.inserisci(slot);
1396 }
1397 //fine modifiche CICALA GIACOMO
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
ntura
1522             morto();
1523             interfaccia.a_capo();
1524             //FINE MODIFICA MICHELE ALBANO
1525             switch (a){
1526                 case 1:
1527                     parlato=false; //aggiunta D'Andria Dresda dal progetto di Gallo in quello d
i Federica Forte
1528                     direzioni(M);
1529                     break;
1530                 case 2:
1531                     prendi();
1532                     break;
1533                 case 3:
1534                     lascia();
1535                     break;
1536                 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 zagara
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
1604     case 49:
1605         consulta(); //inserita la consultazione della guida
1606         //Modifica De Giorgio : fin qui.
1607         break;
1608         //END DAMONE
1609     default: // AZIONI SPECIFICHE
1610         if (!esegui_specifiche(a,M))
1611             interfaccia.scrivi("AZIONE " + a); // test
1612     }
1613 }
```

```

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 Dresden
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'avventura
1648             morto();
1649         interfaccia.a_capo();
1650         //FINE MODIFICA MICHELE ALBANO
1651         switch (a){
1652             case 1:
1653                 parlato = false; //Aggiunta D'Andria Dresden
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:
1683                 macchinadeltempo();
1684                 break;
1685                 //aggiunta D'Andria Dresden
1686             case 99:
1687                 if(parlato==true)
1688                     cout<<"Ci hai gia' parlato"<<endl;
1689                 else{
1690                     az_parla(*pg);
1691                     if(pg->getNome()!="Cid" && pg->getNome()!="Vincent" && pg->getNome()!="Kail") //modifica D'Andria Dresden
1692                 }
```

```

1691                     parlato=true;
1692                 }
1693             break;
1694         //fine aggiunta D'Andria Dresden
1695     case 40:
1696         visualizza_bacheca();
1697         system("cls");
1698         //system("clear"); //linux os //Modificato da ANTONIO PASTORELLI
1699         break;
1700     case 41:
1701         benzina();
1702         break;
1703     case 42:
1704         usa_motorino();
1705         break;
1706         // inizio modifiche zagaria -
1707         - definisce la comunicazione con il personaggio nella storia
1708     case 43:
1709         stringa_risposta = "Hai parlato con il personaggio presente nella stanza.";
1710         //aggiungo alla storia
1711         storia_gioco.insStoria(stringa_comando, stringa_risposta);
1712         az_parla(*pg);
1713         break;
1714         //fine modifiche
1715         //Modifica PMF(agenda)
1716     case 46:
1717         azione_46();
1718         break;
1719     case 47:
1720         azione_47();
1721         break;
1722     case 48:
1723         azione_48();
1724         break;
1725         //Modifica PMF: fin qui.
1726         //START DAMONE
1727         //Modifica Francesco De Giorgio
1728     case 49:
1729         consulta(); //inserita la consultazione della guida
1730         //Modifica De Giorgio : fin qui.
1731         break;
1732         //END DAMONE
1733     default: // AZIONI SPECIFICHE
1734         if (!esegui_specifiche(a,M))
1735             interfaccia.scrivi("AZIONE " + a); // test
1736     }
1737 }

```

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*/,
2213     p31,p32,p33,/*p34,p35,p36,p37,p38,p39,p40,*/p41,p42,p43,/*p44,p45,p46,p4
2214     7,p48,p49,p50,*/p51,p52,p53,/*p54,p55,p56,p57,p58,p59,p60,*/
2215     p61,p62,p63,/*p64,p65,p66,p67,p68,p69,p70,*/p71,p72,p73,/*p74,p75,p76,p7
2216     7,p78,p79,p80,*/p81,p82,p83,/*p84,p85,p86,p87,p88,p89,p90,*/
2217     p91,p92,p93/*,p94,p95,p96,p97,p98,p99,p100*/;
2218
2219     //AGGIUNTA DEI PERSONAGGI PER I DIALOGHI BASILE ANTONIO
2220     Personaggi prof_Mara, prof_Clara, alunno_Davide;
2221
2222     //Luogo 1 1-10
```

```

2221     p1.setNome("Kain");
2222     p1.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2223     ");
2224     p1.setLuogo(1);
2225     insPersonaggi->inserisci(1, p1);
2226     p2.setNome("Tom");
2227     p2.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2228     ");
2229     p2.setOgg(true,51);
2230     p2.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2231     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2232     ile");
2233     p2.setLuogo(1);
2234     insPersonaggi->inserisci(2, p2);
2235     p3.setNome("Jim");
2236     p3.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2237     ");
2238     p3.setLuogo(1);
2239     insPersonaggi->inserisci(3, p3);
2240     p4.setNome("Adam");
2241     p4.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2242     ");
2243     p4.setOgg(true,51);
2244     p4.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2245     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2246     ile");
2247     p4.setLuogo(1);
2248     insPersonaggi->inserisci(4, p4);
2249     p5.setNome("Paul");
2250     p5.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2251     ");
2252     p5.setLuogo(1);
2253     insPersonaggi->inserisci(5, p5);
2254     p6.setNome("Raoul");
2255     p6.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2256     ");
2257     p6.setOgg(true,51);
2258     p6.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2259     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2260     ile");
2261     p6.setLuogo(1);
2262     insPersonaggi->inserisci(6, p6);
2263     p7.setNome("Max");
2264     p7.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2265     ");
2266     p7.setLuogo(1);
2267     insPersonaggi->inserisci(7, p7);
2268     p8.setNome("Sam");
2269     p8.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2270     ");
2271     p8.setOgg(true,51);
2272     p8.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di import
2273     ante","Controlla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di ut
2274     ile");
2275     p8.setLuogo(1);
2276     insPersonaggi->inserisci(8, p8);
2277     p9.setNome("Yan");
2278     p9.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2279     ");
2280     p9.setLuogo(1);
2281     insPersonaggi->inserisci(9, p9);
2282     p10.setNome("Arthur");
2283     p10.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li
2284     ");
2285     p10.setOgg(true,51);

```

```

2268     p10.setFrasiConOggi("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.setFrasiConOggi("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("Getro");
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.setFrasiConOggi("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.setFrasiConOggi("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 pi
lotaggio'");
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 su
d 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 ca
bina");
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 'co
mpartimento 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 pr
eso 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
to' non c'e nulla");
2372 p41.setLuogo(5);
2373 insPersonaggi->inserisci(41, p41);
2374 p42.setNome("Vivi");
2375 p42.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo", "Nell''Armadiet
to' non c'e nulla");

```

```

2376     p42.setOgg(true,54);
2377     p42.setFrasiconOgg1("Apri l' 'Armadietto', potresti trovare qualcosa di utile","La 'Chiave' che hai preso non va bene per questo armadietto, cercane un'altra");
2378     p42.setLuogo(5);
2379     insPersonaggi->inserisci(42, p42);
2380     p43.setNome("Jin");
2381     p43.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo","Nell' 'Armadietto' non c'e nulla");
2382     p43.setOgg(true,54);
2383     p43.setFrasiconOgg1("Togli la 'tata' prima di prendere il camice","Il 'camice' non serve a nulla, lascialo nell' 'armadietto'");
2384     p43.setLuogo(5);
2385     insPersonaggi->inserisci(43, p43);
2386
2387     //luogo 6 51-60
2388     p51.setNome("Cid");
2389     p51.setFrasi("Prendi il 'casco' Potrebbe tornarti utile","Non vedo nulla di utile qui");
2390     p51.setOgg(true,50,51);
2391     p51.setFrasiconOgg1("Prendi la tuta dalla cabina di pilotaggio","La tuta non ti serve, vai direttamente all'esterno dell'astronave");
2392     p51.setFrasiconOgg2("Visto che hai la tuta, puoi uscire all'esterno dell'astronave","lascia il casco prima di uscire dall'astronave");
2393     p51.setLuogo(6);
2394     //modifiche D'Andria Dresden aggiunta dialogo a Cid
2395     Dialogo dialogocid=Dialogo("Cid");
2396     dialogocid=caricaDialoghiDaFile("Cid");
2397     p51.setDialogo(&dialogocid);
2398     //fine modifiche
2399     insPersonaggi->inserisci(51, p51);
2400     p52.setNome("Stainer");
2401     p52.setFrasi("Dopo aver preso il 'casco' cerca la 'tuta'","Ripostati un po sul letto, potrebbe servirti a schiarire le idee");
2402     p52.setLuogo(6);
2403     insPersonaggi->inserisci(52, p52);
2404     p53.setNome("Bruce");
2405     p53.setFrasi("Non uscire all'esterno della nave senza 'casco' e 'tuta'","la 'tuta' e il 'casco' non servono per uscire all'esterno dell'astronave");
2406     p53.setLuogo(6);
2407     insPersonaggi->inserisci(53, p53);
2408
2409     //luogo 7 61-70
2410     p61.setNome("Dom");
2411     p61.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senza 'casco' altrimenti morirai");
2412     p61.setLuogo(7);
2413     insPersonaggi->inserisci(61, p61);
2414     p62.setNome("Gabriel");
2415     p62.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senza 'casco' altrimenti morirai");
2416     p62.setOgg(true,50);
2417     p62.setFrasiconOgg1("Hai il il 'casco', ma ti manca la 'tuta' per poter uscire dall'astronave","hai il 'casco', premi rosso così potrai uscire dall'astronave");
2418     p62.setLuogo(7);
2419     insPersonaggi->inserisci(62, p62);
2420     p63.setNome("Martino");
2421     p63.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senza 'casco' altrimenti morirai");
2422     p63.setOgg(true,50,51);
2423     p63.setFrasiconOgg1("Hai il il 'casco', ma ti manca la 'tuta' per poter uscire dall'astronave","hai il 'casco', premi rosso così potrai uscire dall'astronave");
2424     p63.setFrasiconOgg2("Hai sia il 'casco' che la 'tuta' ora puoi uscire dall'astronave","lascia il 'casco', poi premi rosso così potrai uscire dall'astronave");
2425     p63.setLuogo(7);
2426     insPersonaggi->inserisci(63, p63);
2427
2428     //luogo 8 71-80

```

```

2429     p71.setNome("Samuel");
2430     p71.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo b  
astasse fare il contrario?","Il 'manuale' descrive come attivare il reattore, e se per  
spegnerlo bastasse fare la stessa procedura?");
2431     p71.setLuogo(8);
2432     insPersonaggi->inserisci(71, p71);
2433     p72.setNome("Shain");
2434     p72.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo b  
astasse fare il contrario?","Il 'manuale' descrive come attivare il reattore, e se per  
spegnerlo bastasse fare la stessa procedura?");
2435     p72.setLuogo(8);
2436     insPersonaggi->inserisci(72, p72);
2437     p73.setNome("Zick");
2438     p73.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo b  
astasse fare il contrario?","Il 'manuale' descrive come attivare il reattore, e se per  
spegnerlo bastasse fare la stessa procedura?");
2439     p73.setLuogo(8);
2440     insPersonaggi->inserisci(73, p73);
2441
2442     //luogo 9 81-90
2443     p81.setNome("Lance");
2444     p81.setFrasi("Il 'secondo pilota' e' a nord","si tolga il 'casco' e' sicuro qui");
2445
2446     p81.setLuogo(9);
2447     insPersonaggi->inserisci(81, p81);
2448     p82.setNome("Lauren");
2449     p82.setFrasi("Il 'secondo pilota' e' a nord","si tolga il 'casco' e' sicuro qui");
2450
2451     p82.setLuogo(9);
2452     insPersonaggi->inserisci(82, p82);
2453     p83.setNome("Lucas");
2454     p83.setFrasi("Il 'secondo pilota' e' a nord","si tolga il 'casco' e' sicuro qui");
2455
2456     p83.setLuogo(9);
2457     insPersonaggi->inserisci(83, p83);
2458
2459     //luogo 10 91-100
2460     p91.setNome("Andre'");
2461     p91.setFrasi("Soccorra il 'secondo pilota', e nei guai","il 'secondo pilota' e' and  
ato perso nello spazio, e' inutile cercarlo");
2462     p91.setLuogo(10);
2463     insPersonaggi->inserisci(91, p91);
2464     p91.setNome("George");
2465     p91.setFrasi("Soccorra il 'secondo pilota', e nei guai","il 'secondo pilota' e' and  
ato perso nello spazio, e' inutile cercarlo");
2466     p91.setLuogo(10);
2467     insPersonaggi->inserisci(92, p91);
2468     p91.setNome("Oscar");
2469     p91.setFrasi("Soccorra il 'secondo pilota', e nei guai","il 'secondo pilota' e' and  
ato perso nello spazio, e' inutile cercarlo");
2470     p91.setLuogo(10);
2471     insPersonaggi->inserisci(93, p91);
2472
2473     //aggiunta modifiche D'Andria Dresden al progetto di Federica Forte caricamento dial  
oghi da file
2474     scrivistatosofile("Cid",1);
2475     scrivistatosofile("Vincent",1);
2476     scrivistatosofile("Kail",1);
2477     //fine modifiche
2478 }
```

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 fogli
2665 a del dialogo si reimposta lo stato a 1
2666                 }
2667             } while(!dialogofinito);
2668         }
2669         else{
2670             if(pg.getIfNec())
2671             {
2672                 ris_controllo=controlla();
2673                 if(ris_controllo==0)
2674                     pg.SelezionaFrase2();
2675                 else if(ris_controllo==1)
2676                     pg.SelezionaFrase1();
2677                 else
2678                     parla(pg);
2679             }
2680         }
2681     }
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 template <class tipoelem>
15 class Albero{
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 primofiglio(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 cancottoalbero(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     BinAlbero<tipoelem> tree;
63     //METODO CHE COPIA UN ALBERO PASSATO PARTENDO DAL NODO SINISTRO
64     void copia_albero_sx(const BinAlbero<tipoelem>&, const nodo&, nodo);
65     //METODO CHE COPIA L'ALBERO PASSATO PARTENDO DAL NODO DESTRO
66     void copia_albero_dx(const BinAlbero<tipoelem>&, const nodo&, nodo);
67 };
68
69 template <class tipoelem>
70 Albero<tipoelem>::Albero(){
71     creaalbero();
72 }
73
74 template <class tipoelem>
75 Albero<tipoelem>::Albero(const Albero& a){
76     if(!a.tree.binalberovuoto())
77     {
78         tree.insbinradice();
79     }
80 }

```

```

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.sinistruvoto(a.tree.binradice()))
80         {
81             if(!a.tree.destrovuto(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     template <class tipoelem>
91     Albero<tipoelem>::~Albero(){
92         //RICHIAMA AUTOMATICAMENTE IL DISTRUTTORE
93         //DELL'ALBERO BINARIO
94     }
95
96     template <class tipoelem>
97     void Albero<tipoelem>::creaalbero(){
98         //RICHIAMA AUTOMATICAMENTE IL COSTRUTTORE
99         //DELL'ALBERO BINARIO
100    }
101
102    template <class tipoelem>
103    bool Albero<tipoelem>::alberovuoto() const{
104        return tree.binalberovuoto();
105    }
106
107    template <class tipoelem>
108    void Albero<tipoelem>::insradice(){
109        tree.insbinradice();
110    }
111
112    template <class tipoelem>
113    typename Albero<tipoelem>::nodo Albero<tipoelem>::radice() const{
114        return tree.binradice();
115    }
116
117    template <class tipoelem>
118    typename Albero<tipoelem>::nodo Albero<tipoelem>::padre(typename Albero<tipoelem>::nodo
119 n){
120        return tree.binpadre(n);
121    }
122
123    template <class tipoelem>
124    bool Albero<tipoelem>::foglia(typename Albero<tipoelem>::nodo n) const{
125        if(tree.sinistruvoto(n))
126            return true;
127        else
128            return false;
129    }
130
131    template <class tipoelem>
132    typename Albero<tipoelem>::nodo Albero<tipoelem>::primofiglio(typename Albero<tipoelem>
133 ::nodo n) const{
134        return tree.figliosinistro(n);
135    }
136
137    template <class tipoelem>
138    bool Albero<tipoelem>::ultimofratello(typename Albero<tipoelem>::nodo n) const{
139        if(tree.destrovuto(n))
140            return true;

```

```

139     else
140         return false;
141     }
142
143     template <class tipoelem>
144     typename Albero<tipoelem>::nodo Albero<tipoelem>::succfratello(typename Albero<tipoelem>::nodo n) const{
145         return tree.figliodestro(n);
146     }
147
148     template<class tipoelem>
149     void Albero<tipoelem>::copia_albero_sx(const BinAlbero<tipoelem>& other, const nodo& radice_sottoalbero, nodo nodo_padre)
150     {
151         tree.insfigliosinistro(nodo_padre);
152         nodo sx = tree.figliosinistro(nodo_padre);
153         tree.scrivinodo(sx, other.legginodo(radice_sottoalbero));
154
155         if(!other.sinistruvoto(radice_sottoalbero))
156             copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), sx);
157         if(!other.destrovuto(radice_sottoalbero))
158             copia_albero_dx(other, other.figliodestro(radice_sottoalbero), sx);
159     }
160
161     template <class tipoelem>
162     void Albero<tipoelem>::copia_albero_dx(const BinAlbero<tipoelem>& other, const nodo& radice_sottoalbero, nodo nodo_padre)
163     {
164         tree.insfigliodestro(nodo_padre);
165         nodo dx = tree.figliodestro(nodo_padre);
166         tree.scrivinodo(dx, other.legginodo(radice_sottoalbero));
167
168         if(!other.sinistruvoto(radice_sottoalbero))
169             copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), dx);
170         if(!other.destrovuto(radice_sottoalbero))
171             copia_albero_dx(other, other.figliodestro(radice_sottoalbero), dx);
172     }
173
174     template <class tipoelem>
175     void Albero<tipoelem>::insprimosottoalbero(typename Albero<tipoelem>::nodo n, Albero& a)
176     {
177         nodo temp;
178         if(tree.sinistruvoto(n)){
179             copia_albero_sx(a.tree,a.tree.binradice(),n);
180         }
181         else{
182             Albero<tipoelem> T;
183             T.insradice();
184             temp=T.radice();
185             T.scrivinodo(temp,tree.legginodo(tree.figliosinistro(n)));
186
187             if(tree.sinistruvoto(tree.figliosinistro(n))){
188                 if(!tree.destrovuto(tree.figliosinistro(n))){
189                     copia_albero_dx(tree,tree.figliodestro(tree.figliosinistro(n)),temp);
190                 }
191             }
192             else{
193                 copia_albero_sx(tree,tree.figliosinistro(tree.figliosinistro(n)),temp);
194             }
195
196             tree.cancsottobinalbero(tree.figliosinistro(n));
197
198             copia_albero_sx(a.tree,a.tree.binradice(),n);
199             temp=tree.figliosinistro(n);
200             copia_albero_dx(T.tree,T.radice(),temp);
201         }
202     }

```

```

201 }
202
203
204 template <class tipoelem>
205 void Albero<tipoelem>::insottoalbero(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             cancsottoalbero(tree.figliodestro(n));
231
232             copia_albero_dx(a.tree,a.radice(),n);
233             temp=tree.figliodestro(n);
234             copia_albero_dx(T.tree,T.radice(),temp);
235         }
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 cancossottobinalbero(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
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
61     private:
62         Nodo radice;
63     };
64
65     template <class TIPOETICHETTA>
66     BinAlbero<TIPOETICHETTA>::BinAlbero(){
67         creabinalbero();
68     }
69
70     template <class TIPOETICHETTA>
71     BinAlbero<TIPOETICHETTA>::~BinAlbero(){
72         /* if(radice!=nil)

```

```

68         cancsottobinalbero(binradice()));
69     */
70 }
71
72 template <class TIPOETICHETTA>
73     void BinAlbero<TIPOETICHETTA>::creabinalbero(){
74     radice=nil;
75 }
76
77 template <class TIPOETICHETTA>
78     bool BinAlbero<TIPOETICHETTA>::binalberovuoto()const{
79         if(radice==nil)
80             return true;
81         else
82             return false;
83     }
84
85 template <class TIPOETICHETTA>
86     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binradice()const{
87
88         return radice;
89     }
90
91 template <class TIPOETICHETTA>
92     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binpadre(Nodo padre)const{
93         return padre->getPadre();
94     }
95
96 template <class TIPOETICHETTA>
97     bool BinAlbero<TIPOETICHETTA>::sinistrovuoto(Nodo sx)const{
98         if(sx->getFiglioSx()==nil)
99             return true;
100        else
101            return false;
102    }
103
104 template <class TIPOETICHETTA>
105     bool BinAlbero<TIPOETICHETTA>::destruovoato(Nodo dx)const{
106         if(dx->getFiglioDx()==nil)
107             return true;
108         else
109             return false;
110    }
111
112 template <class TIPOETICHETTA>
113     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliosinistro(Nodo sx)const{
114         return sx->getFiglioSx();
115     }
116
117 template <class TIPOETICHETTA>
118     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliodestro(Nodo dx)const{
119         return dx->getFiglioDx();
120     }
121
122 template <class TIPOETICHETTA>
123     void BinAlbero<TIPOETICHETTA>::costrbinalbero(BinAlbero<TIPOETICHETTA> &A,BinAlbero<TIPOETICHETTA> &B){
124         radice=new Nodo;
125         radice->setPadre(nil);
126
127         if(!A.binalberovuoto()){
128             radice->setFiglioSx(A.binradice());
129             radice->getFiglioSx()->setPadre(radice);

```

```

130 }
131     else
132         radice->setFiglioSx(nil);
133
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(!destruovoto(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
169 template <class TIPOETICHETTA>
170     TIPOETICHETTA BinAlbero<TIPOETICHETTA>::legginodo(Nodo n)const{
171         return n->getEtichetta();
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
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

```

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     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    template <class TIPOETICHETTA>
109        Cella_Binalbero<TIPOETICHETTA>::Cella_Binalbero(const Cella_Binalbero& c){
110            etichetta=c.getEtichetta();
111            FiglioDx=c.getFiglioDx();
112            FiglioSx=c.getFiglioSx();
113            Padre=c.getPadre();
114        }
115
116    template <class TIPOETICHETTA>
117        void Cella_Binalbero<TIPOETICHETTA>::operator=(const Cella_Binalbero& c ){
118            etichetta=c.getEtichetta();
119        }

```

```

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
7 Dialogo::Dialogo(const Dialogo& d){
8     personaggio = d.personaggio;
9     stato = d.stato;
10    dialogo = Albero<Domanda>(d.dialogo);
11 }
12
13 Dialogo::Dialogo(string nomepersonaggio){
```

```

14     this->personaggio=nomepersonaggio;
15     this->stato=1;
16 }
17
18 void Dialogo::setstato(int numero){
19     this->stato=numero;
20 }
21
22 int Dialogo::getstato(){
23     return (this->stato);
24 }
25 string Dialogo::leggiPersonaggio(){
26     return (this->personaggio);
27 }
28
29 void Dialogo::inseriscidomanda(int codice, string frase){
30     if (codice==1)
31         inseriscipresentazione(frase);
32     else
33     {
34         Albero<Domanda>::nodo indicenodo=Albero<Domanda>::nil;
35         int codicedacercare=0;
36         int modulocodice=codice%10;
37         if(modulocodice==1){
38             codicedacercare=(codice-1)/10;
39             indicenodo = trovanodo(dialogo.radice(),codicedacercare);
40             inserisciprimaopzione(indicenodo,codice,frase);
41         }
42         else{
43             codicedacercare=codice-1;
44             indicenodo = trovanodo(dialogo.radice(),codicedacercare);
45             inserisciopzionessuccessiva(indicenodo,codice,frase);
46         }
47     }
48 }
49
50 Albero<Domanda>::nodo Dialogo::trovanodo(Albero<Domanda>::nodo indice, int codice){
51     Albero<Domanda>::nodo nodo;
52     Albero<Domanda>::nodo nododaritornare=Albero<Domanda>::nil;
53
54     if(codice==dialogo.legginodo(indice).leggicodice())
55         nododaritornare=indice;
56
57     else{
58         if(!dialogo.foglia(indice)){
59             nodo=dialogo.primofiglio(indice);
60             while(!dialogo.ultimofratello(nodo) && nododaritornare==Albero<Domanda>::ni
61             l){
62                 nododaritornare=trovanodo(nodo,codice);
63                 nodo=dialogo.succfratello(nodo);
64             }
65             if (nododaritornare==Albero<Domanda>::nil)
66                 nododaritornare=trovanodo(nodo,codice);
67         }
68     }
69     return nododaritornare;
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     string 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));
139 }
```

5.6.4.3.3 Dialoghi.txt

-Cid|1*Ciao, sono Cid!!\$\$Credi in Dio?\$1:s\$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!$Co
sa 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$0:Interro
mpi dialogo$#
-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$0:interrompi dialo
go$#
-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.$#
-
Vincent|1113*Male, prova il loro gelato alla panna e' delizioso, ricordati di usare il cou
pon!$#
-
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$0:interrompi dialo
go$#
-
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 p
iu' pallida idea$0:interrompi dialogo$#
-Kail|1121*Ottimo, corri a prenderla e recati al pronto soccorso!!!$#
-
Kail|1122*Come hai fatto a non vederla?! E' nella tua cabina! Prendila e vai al pronto soc
corso!!!$#
?

```

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_
utente=="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 TRIPLICARE 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)  
//sovraffaccio 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
```

```
List<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
canzoneInRiproduzione = 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("SUCESSIVA") == 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, via 6,2,2,**via 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.");
    }
}

else if(input=="no" || input=="NO" || input=="No" || input=="n" || input=="N" || input=="nO"){
    interfaccia.scrivi("Hai deciso di non prenotare alcun biglietto");
    interfaccia.a_capo();
    //FACCIO UN FUORI CODA PER "SVUOTARE" LA CODA
    i=0;
}

```

```

while(i<=casuale){

    i++;

    c.fuoricoda();

}

}

else{

    interfaccia.scrivi("non capisco.");

    interfaccia.a_capo();

}

}

}

else{

    interfaccia.scrivi("#####");

    interfaccia.scrivi("#####");

    interfaccia.scrivi("#### non puoi accedere alla biglietteria perche' non hai soldi con cui pagare ####");

    interfaccia.scrivi("#### per accedere alla biglietteria serve avere: -un portafoglio ####");

    interfaccia.scrivi("#### grazie e arrivederci ####");

    interfaccia.scrivi("#####");

}

}

```

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(biglietto==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(biglietto==false{

    interfaccia.scrivi("Per salire sul treno e' necessario acquistare un biglietto");

    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 biglietteria
void azione_116();//leggi giornale
void azione_117();//Sali sul treno

```

//FINE modifiche SALVATORE VESTITA

5.9.5 Biglietto.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

5.10.3.2 Inserimento azioni

```
//INIZIO modifiche VINCENZO GIANNUZZO
vocabolario.inserisci("panca", 57);
vocabolario.inserisci("tapis", 47);

//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 Palestra, 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 rimanete in base alle ripetizioni effettuate.

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

```
void Astro::azione_118() //usa panca
{
    interfaccia.scrivi("Quante ripetizioni vuoi fare ?");
    int rip;
    string risp;
    getline(cin, risp);
    stringstream(risp) >> rip;
    Palestra::usaPanca(sFisico, rip);
}
```

rimanente per finire il gioco.

L'azione 120 invoca la funzione di usare il terminale, contenuta sempre all'interno della classe "Palestra".

L'azione 121, chiama la funzione di accedere alle schede, inglobata all'interno della classe "Palestra".

```
void Astro::azione_119() // usa tapis
{
    interfaccia.scrivi("Per quanti minuti vuoi correre ?");
}

void Astro::azione_120() //usa/guarda terminale
{
}

void Astro::azione_121()
{
    Palestra::usaSchede(sFisico);
}
```

5.10.4 Astro.h

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

```
//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
```

perso e le probabilità essere feriti.

Negli altri casi “tempo_perso” non viene aggiornata perché la sazietà è troppo bassa e la penalità viene sottratta direttamente da “tempo”.

```
float tempo_perso = 0; //Modifica GIANNUZZO
if (Salute.GetStatoSalute() <= 40)
{
    tempo -= Salute.GetPenalita();
}
else //La penalità non viene ridotta se la salute è troppo bassa
{
    tempo_perso += Salute.GetPenalita();
}
tempo_perso+= 2; //Modifica GIANNUZZO
if (sazio <= 60 && sazio >= 50)
    interfaccia.scrivi("\nE' tutto il giorno che non mangio, do-vrei fare una pausa...");
}
```

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;
```

```
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;
```

5.10.7 StatoFisico.h

Classe che implementa le caratteristiche fisiche del personaggio

```

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;

}

```

5.10.8 StatoFisico.cpp

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.



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++;

}

indiceDisponibilita = 31;

while(indiceDisponibilita<=31 && !disponibilita100)

{

```

```

if(slotmachine.appartiene(indiceDisponibilita))
disponibilita100 = true;
indiceDisponibilita++;
}

cout<< " 5 euro: "; disponibilita5 ? cout<<"DISPONIBILE"<<endl : cout<<"ESAURITO"<<endl;
cout<< " 10 euro: "; disponibilita10 ? cout<<"DISPONIBILE"<<endl : cout<<"ESAURITO"<<endl;
cout<< " 20 euro: "; disponibilita20 ? cout<<"DISPONIBILE"<<endl : cout<<"ESAURITO"<<endl;
cout<< " 50 euro: "; disponibilita50 ? cout<<"DISPONIBILE"<<endl : cout<<"ESAURITO"<<endl;
cout<<"100 euro: "; disponibilita100 ? cout<<"DISPONIBILE"<<endl : cout<<"ESAURITO"<<endl;
interfaccia.scrivi("Cosa desideri fare?");
interfaccia.scrivi("");
interfaccia.scrivi("1. Gioca (spenderai 1 Euro)");
interfaccia.scrivi("2. Esci");
cin>>risposta;
bool nonvalido=false;
while(sentinella && nonvalido==false)
{
fflush(stdin);
if(risposta == "gioca" || risposta == "Gioca" || risposta == "g" || risposta == "G" || risposta == "1")
{
//moneta
//cout<<"soldi"<<portafoglio.get_contanti();
if(portafoglio.get_contanti()>0){
float soldi=portafoglio.get_contanti()-1.00;
portafoglio.set_contanti(soldi);

cout<<"soldi attuali= "<<portafoglio.get_contanti();
srand((unsigned)time(NULL));
numeroEstratto = (rand() % 100) +1;
if(numeroEstratto>=1 && numeroEstratto<=31)
{
interfaccia.scrivi("");
}
}
}
}

```

```

if(numeroEstratto>=1 && numeroEstratto<=16 && disponibilita5)
{
    interfaccia.scrivi("Congratulazioni! Hai vinto 5 euro!");
    soldi=portafoglio.get_contanti()+5.00;
    portafoglio.set_contanti(soldi);
    oggetti.set_luogo(31+numeroEstratto,0);
    slotmachine.cancella(numeroEstratto);
}

else if(numeroEstratto>=17 && numeroEstratto<=24 && disponibilita10)
{
    interfaccia.scrivi("Congratulazioni! Hai vinto 10 euro!");
    soldi=portafoglio.get_contanti()+10.00;
    portafoglio.set_contanti(soldi);
    oggetti.set_luogo(31+numeroEstratto,0);
    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;  
  
svuotalinsieme(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';
```

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.             "\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. |"; cout << "\n| progettato per il viaggio interstellare | scienziati (Freeman Dyson e il
10. o |"; cout << "\n| cioe' in grado di raggiungere sistemi | Progetto Orione) hanno discuss
11. e |"; cout << "\n| stellari diversi da quello di partenza. | diverse ipotesi ingegneristiche
12. |"; cout << "\n| La fantascienza (in particolare il filone | riguardanti la possibilita' di
13. aggi |"; cout << "\n| della space opera) abbonda di storie che | intraprendere in futuro dei vi
14. |"; cout << "\n| descrivono questo genere di veicoli | interstellari.
15. |"; cout << "\n| degli anni molti autori hanno descritt |"
16. |"; cout << "\n| e nel corso astronavi di varie forme: |"
17. |"; cout << "\n| in alcuni anime l'astronave principale |"
18. |"; cout << "\n| corazzata giapponese della seconda guerra|"
```

```

19.         cout << "\n| aveva la forma di una mondiale, nella |"
|";
20.         cout << "\n| Guida galattica per gli autostoppisti ci |"
|";
21.         cout << "\n| sono astronavi a forma di mattoni |"
|";
22.         cout << "\n| giallim mentre le classiche astronavi |"
|";
23.         cout << "\n| alien degli anni cinquanta sono i celebri|"
|";
24.         cout << "\n|dischi volanti. |"
|";
25.         cout << "\n| -----|-----|-----|-----|-----|-----|-----|"
----|";
26.         system("pause");
27.         break;
28.     case 91:
29.         cout << "\n| -----|-----|-----|-----|-----|-----|-----|"
----|";
30.         cout << "\n|          EQUIPAGGIO"
|";
31.         cout << "\n| Questa e' una lista delle persone che | scritti in corsivo.Essi non ha"
nno |";
32.         cout << "\n| hanno partecipato come equipaggi | specifico ruolo ma sono inseriti"
ti |";
33.         cout << "\n| divisi in ordine | specialisti di carico per ragioni"
oni |";
34.         cout << "\n| cronologico rispetto alla missione. | di spazio."
|";
35.         cout << "\n| Abbreviazioni usate: | STS-61-"
A e' l'unico volo, lanciato|";
36. do |";
37.         cout << "\n| PC = Comandante del carico utile | nel 1985, che ha portato a bordo"
|";
38.         cout << "\n| MSE = Ingegnere di volo dell'USAF | equipaggio maggiore di sette"
|";
39.         cout << "\n| Mir = Lanciato come equipaggio della | astronauti."
|";
40.         cout << "\n| ISS = Lanciato come equipaggio della |"
|";
41.         cout << "\n| Stazione Spaziale Internazionale (ISS). |"
|";
42.         cout << "\n| I nomi degli astronauti di ritorno dalla |"
|";
43.         cout << "\n| Mir o dall'ISS sullo Space Shuttle sono |"
|";

```

```

44.         cout << "\n| ----- |----- |----- |----- |----- |
45.         ----- |";
46.         system("pause");
47.     }
48. }
```

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.suc-
clist(a(p)) {}
16.     Oggetto oggettoPreso = oggetti.get_oggetto(og);
17.     int codOggetto = oggettoPreso.get_codice();
18.     if (codOggetto >= 90) {
19.         libriInPrestito.inslista(codOggetto, p);
20.     }
21.     system("cls");
22.     cout << "Hai preso in prestito " << oggettoPreso.get_nome() << ".\n";
23.     oggetti.set_luogo(og, 0);
24.     j++;
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, -                           interfaccia.scrivi("Si e' perso nello spazio.");
4.     99);           stringa_risposta = "si e' perso nello spazio!!"; //Modifica PMF(storia)
5.     interfaccia.scrivi(stringa_risposta); //Modifica PMF(storia)
6.     storia_gioco.insStoria(stringa_comando , stringa_risposta); //Modifica PMF(storia)
7.     } else           lasciato = false;                      return lasciato;*/ // modifica Gallone
8.     bool lasciato = true; //Inizio Modifica Gallone - aggiorno il metodo per integrare le nuove funzionalità
9.     riferimento_tasca = tasca.primolista();
10.    riferimento_zaino = zaino_frigo.primolista();
11.    if (luogo_attuale == 9 || luogo_attuale == 10 || luogo_attuale == 11 || (luogo_attuale == 7 &
12. & parete_stagna_aperta == 1)) {
13.        if (og >= 35 && og <= 37) {
14.            while (!tasca.finelista(riferimento_tasca)) {
15.                if (oggetti.get_oggetto(tasca.leggilista(riferimento_tasca)).get_codice() == ogge-
tti.get_oggetto(og).get_codice()) tasca.canclista(riferimento_tasca);
16.            else riferimento_tasca = tasca.succlista(riferimento_tasca);
17.        }
18.        } else if (og >= 25 && og <= 34) {

```

```

13.         while (!zaino_frigo.finelista(riferimento_zaino)) {
14.             if (oggetti.get_oggetto(zaino_frigo.legglista(riferimento_zaino)).get_codice() ==
15.                 oggetti.get_oggetto(og).get_codice()) zaino_frigo.canlista(riferimento_zaino);
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 SUG-
    LI 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.suc-
    clista(p)) {
28.             if (libriInPrestito.legglista(p) == codOggetto) {
29.                 scadenze[j] = 30;
30.                 libriInPrestito.canlista(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.suclist(p);
34.             j++;
35.         }
36.     }
37. }
38. }
39. } else lasciato = false;
40. return lasciato;
41. }

```

5.12.6.3scadenza()

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.succl
ista(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 " << "
13. \n                     Corri in Biblioteca!\n";
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.                 }
24.             }
25.         }
26.         libriInPrestito.canclista(p);
27.         tempo = tempo - 10; //PENALITA'
28.     }
29.     j++;
30. }
31. }
```

5.12.6.4aggiorna_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. }
```

5.12.6.5ciak(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.            }
39.            l=mappa.get_luogo(luogo_attuale);
40.            if(!Luogo_p.pilavuota() && (Codice_luogo.leggipila()!=luogo_attuale))
41.                //MODIFICA POLIMENA
42.                {
43.                    cout<<"Provieni dal luogo: "<< Codice_luogo.leggipila() << ":" << Luogo
44.                    _p.leggipila(); // MODIFICA POLIMENA
45.                }
45.                Luogo_p.inpila(mappa.get_nome_luogo(luogo_attuale));
46.                //MODIFICA POLIMENA
46.                Codice_luogo.inpila(luogo_attuale);
47.                //MODIFICA POLIMENA
47.
48.                if(luogo_attuale == 7)
49.                {
50.                    if(oggetti.get_oggetto(4).get_luogo() != 0 || oggetti.get_oggetto(11).g
et_luogo()!= 0)
51.                        bacheca.NuovoMessaggioGioco("*non uscire fuori dall'astronave se no
n hai l'equipaggiamento da astronauta");
51.                    //else
```

```

52.          //bacheca.CancellaMessaggioGioco("*non uscire fuori dall'astronave se non hai l'equipaggiamento da astronauta");
53.      }
54.      // MODIFICA D-R(Colturi):Luoghi a Pagamento-----
55.      if(luogo_attuale > 15) //|
56.      {
57.          //|
58.          Luogo l = M.get_luogo(luogo_attuale); //|
59.          //|
60.      } //|
61. // FINE MODIFICA D-R-----|
62. interfaccia.disegna_scena(1);
63.
64. personaggio_entra(); // modifica zagarria
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. while (riparti);
79. }

```

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 INTERMEDIATE
45.         interfaccia.scrivi("Fatto.");
46.     } // modifica Gallone - spostamento parentesi per correzione di un errore che si generava nel
1 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  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
if (libriInPrestito.legglista(p) == codOggetto)  
{  
    scadenze[libriInPrestito.legglista(p)] = 30;  
    libriInPrestito.canclista(p);  
    oggetti.set_luogo(og, luogo_attuale);  
    cout << "Fatto.\nhai restituito "  
        << oggetti.get_oggetto(og).get_nome() << ".\n";  
    scadenze[j]=30;  
    p = libriInPrestito.succlista(p);  
    j++;  
}
```

```
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
if (libriInPrestito.legglista(p) == codOggetto)  
{  
    scadenze[j] = 30;  
    libriInPrestito.canclista(p);  
    oggetti.set_luogo(og, luogo_attuale);  
    cout << "Fatto.\nhai restituito "  
        << oggetti.get_oggetto(og).get_nome() << ".\n";  
    p = libriInPrestito.succlista(p);  
}
```

```
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** (NNSEOOUUDD)
- **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.

```

File Modifica Formato Visualizza ?
di nuovo a Roma
A Roma
Nuovamente a Roma
Ancora a Roma

```

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
è depositato nel bagagliaio

                }

            }

            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 FOLGORE 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){
```

```
}
```

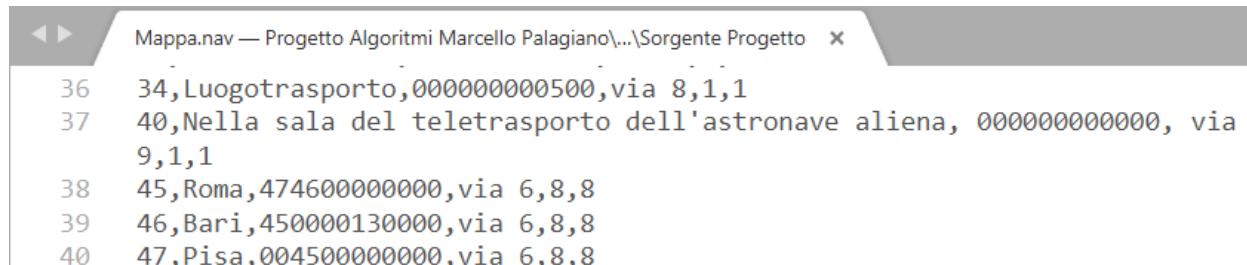
```
Automobile::~Automobile(){
```

```
}
```

5.14 MODIFICA CODICI LUOGHI – PALAGIANO MARCELLO

5.14.1 Modifica file “Mappa.nav”

Il codice nel progetto base1906 era il seguente:



```
Mappa.nav — Progetto Algoritmi Marcello Palagiano\...\Sorgente Progetto x
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:



```
Mappa.nav — Progetto ASD Palagiano Marcello\Progetto ASD Palagiano x
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,00003900004,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:

```
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  
-
```

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 modifche DAVIDE MANTELLINI
```

```
//INIZIO modifche GIACOMO CICALA -- Palagiano Marcello: modifica codice luogo da 36 a 35  
azioni.inserisci(3500290086, 125); //usa slotmachine  
//FINE modifche GIACOMO CICALA
```

```
//inizio modifche 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 modifche 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
```

5.14.6.1 *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.finelist(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"
        "\n\nin biblioteca");

    interfaccia.scrivi("\n4) Si prega infine di fare silenzio!");

    interfaccia.scrivi("\n\nSaluti dal Bibliotecario... \n");
}

```

5.14.6.2 *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.finelista(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.finelista(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.preclista(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);

}

```

5.15.8.6.10 Azione 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!");

        }

    }

}

```

```

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”, LUOGO “CASERMA” E OGGETTO CASSAFORTE

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.setFras("Il carabiniere ti ha perquisito e ha trovato in tuo possesso un oggetto rubato, sei in a
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: "<< carabinier
}
```

```

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;
    }
}

```

```

1319 void Gioco::perquisizione_giocatore(int &luogo_attuale){
1320
1321     if (oggetti.get Oggetto(118).get_rubato() == true || oggetti.get Oggetto(73).get_rubato() == true || oggetti.get Oggetto(49).get_rubato() == true)
1322     {
1323         interfaccia.scrivi("Il carabiniere ti ha perquisito e ha trovato in tuo possesso un oggetto rubato, sei in arresto!");
1324         interfaccia.a_capo();
1325         luogo_attuale = mappa.nodo_da_nome_etichetta("Nella caserma").getNodo() + 1; //modifica Davide Rano
1326         interfaccia.scrivi("Puoi uscire dalla caserma fra 30 secondi.");
1327         SLEEP (30);
1328         interfaccia.scrivi("Ora sei libero.");
1329
1330         oggetti.set_rubato(118, false);
1331         oggetti.set_rubato(73, false);
1332         oggetti.set_rubato(49, false);
1333
1334         //inizio modifica Davide Rano
1335         if(oggetti.get_luogo(118) == 0) { // <-- se sta nell'inventario (quindi rubato)
1336             cassaforte.inserisci(118);
1337             oggetti.set_luogo(118, -9999);
1338         }
1339
1340         if(oggetti.get_luogo(73) == 0){
1341             cassaforte.inserisci(73);
1342             oggetti.set_luogo(73, -9999);
1343         }
1344         if(oggetti.get_luogo(49) == 0) {
1345             cassaforte.inserisci(49);
1346             oggetti.set_luogo(49, -9999);
1347         }
1348         // fine modifica Davide Rano
1349     }
1350     else
1351     {
1352         interfaccia.scrivi("Il carabiniere ti ha perquisito e non ha trovato nessun oggetto rubato in tuo possesso.");
1353     }
1354     interfaccia.a_capo();
1355 }

```

Apri cassaforte

```

6663 void Astro::azione_156()
6664 {
6665     int combinazione;
6666
6667     if(!cassaforte.get_aperta()) {
6668         interfaccia.scrivi("Il lucchetto della cassaforte ha 4 quadranti, che vanno da 0 a 9 ciascuno");
6669
6670         combinazione = interfaccia.leggi_intero("Inserire la combinazione: ");
6671         interfaccia.a_capo();
6672
6673     if(combinazione == cassaforte.get_combinazione()) {
6674         interfaccia.scrivi("Click... La cassaforte si e' aperta!!!");
6675         cassaforte.set_aperta(true);
6676
6677         for (int i=1; i <= cassaforte.get_n_oggettiSequestrati(); i++)
6678             oggetti.set_luogo(cassaforte.svuotaCassaforte(), mappa.nodo_da_nome_etichetta("Nella caserma").getNodo() + 1);
6679     }
6680
6681     else
6682         interfaccia.scrivi("La combinazione e' sbagliata..");
6683 }
6684 else
6685     interfaccia.scrivi("La cassaforte era già aperta");
6686
6687
6688
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
6715
6716
6717
6718
6719
6720
6721
6722
6723
6724
6725
6726
6727
6728
6729
6730
6731
6732
6733
6734
6735
6736
6737
6738
6739
6740
6741
6742
6743
6744
6745
6746
6747
6748
6749
6750
6751
6752
6753
6754
6755
6756
6757
6758
6759
6760
6761

```

```

1071     oggetti.inserisci(Oggetto("una cella",121,-46));
1072     oggetti.inserisci(Oggetto("una vetrina",122,-46));
1073     oggetti.inserisci(Oggetto("una cassaforte",123,-46));
1074     ...
489      vocabolario.inserisci("cella", 201);
490      vocabolario.inserisci("cassaforte", 203);

812      azioni.inserisci(4600100201, 153);
813      azioni.inserisci(4600100019, 154);
814      azioni.inserisci(4600100203, 155);
815      azioni.inserisci(4600220203, 156);
     ...

7192      case 153: azione_153(); break;
7193      case 154: azione_154(); break;
7194      case 155: azione_155(); break;
7195      case 156: azione_156(); break;

```

Le azioni guarda cella, guarda vetrina, guarda cassaforte e apri cassaforte.

Nel file Astro.h è stato aggiunto:

```

303      void azione_153();
304      void azione_154();
305      void azione_155();
306      void azione_156();

```

Sono stati aggiunti i file Arma.h, Armi.h, Caserma.h, Cassaforte.h

```
1  ifndef ARMA_H
2  define ARMA_H
3  #include <iostream>
4  #include <string>
5  #include <stdlib.h>
6
7
8  using namespace std;
9
10 class Arma
11 {
12 public:
13     Arma();
14     Arma(string nome, string descrizione);
15     ~Arma();
16
17     string get_nome() const;
18     void set_nome(string);
19
20     string get_descrizione () const;
21     void set_descrizione (string);
22
23     string stampa_Arma() const;
24
25 private:
26
27     string nome;
28     string descrizione;
29
30 };
31
32
33
34 #endif /* ARMA_H */
35
```

```
#ifndef ARMI_H
#define ARMI_H

#include <iostream>
#include <string>
#include <stdlib.h>
#include "Lista.h"
#include "servizioLista.h"
#include "Arma.h"

using namespace std;

class Armi
{
public:
    Armi () ;

    void inserisci (string nome, string desc);
    int get_n_Armi () ;

    string get_Arma (int) ;
    string get_nome_Arma (int) ;
    string get_descrizione_Arma (int) ;

    void elenca_Armi () ;

private:
    Lista<Arma> l_Armi;
    int n_Armi;
    Lista<Arma>::posizione pos;
};

#endif /* Armi_H */
```

file Arma.h

```
1      #ifndef CASERMA_H_INCLUDED
2      #define CASERMA_H_INCLUDED
3
4      #include <iostream>
5
6      using namespace std;
7      #include <string>
8      #include "Lista.h"
9      #include "Armi.h"
10
11     class Caserma {
12
13     public:
14         Caserma();
15         void mostraArmi();
16
17     private:
18
19         Armi armiDepositate;
20     };
21
22     #endif // CASERMA_H_INCLUDED
23
```

file Armi.h

file Caserma.h

```
2 #define CASSAFORTE_H
3
4 #include <iostream>
5 #include <string>
6 #include <stdlib.h>
7 #include "Pila.h"
8 #include "Oggetti.h"
9
10 using namespace std;
11
12 class Cassaforte
13 {
14 public:
15
16     Cassaforte();
17     void inserisci(int);
18     int svuotaCassaforte();
19
20     int get_n_oggettiSequestrati();
21     int get_combinazione();
22     bool get_aperta();
23     void set_aperta(bool);
24
25 private:
26
27     int n_oggetti_sequestrati;
28     int combinazione;
29     bool aperta;
30
31     Pila<int> sequestrati;
32
33 };
34
35 #endif /* CASSAFORTE_H */
36
```

file Cassaforte.h

Sono stati aggiunti i file Armi.cpp, Cassaforte.cpp, Arma.cpp, Caserma.cpp

```

1 #include "Cassaforte.h"
2
3
4 using namespace std;
5
6 Cassaforte::Cassaforte()
7 {
8
9     n_oggetti_sequestrati = 0;
10    combinazione = 2021;
11    aperta = false;
12    sequestrati.creapila();
13 }
14
15 void Cassaforte::inserisci(int seq)
16 {
17     sequestrati.inpila(seq);
18     n_oggetti_sequestrati++;
19     aperta = false;
20 }
21
22 int Cassaforte::svuotaCassaforte()
23 {
24     int i = sequestrati.leggipila();
25     sequestrati.fuoripila();
26     // set_aperta(true);
27     aperta = true;
28     return i;
29 }
30
31 int Cassaforte::get_n_oggettiSequestrati()
32 {
33     return n_oggetti_sequestrati;
34 }
35

```

file Cassaforte.cpp

```

1 #include "Caserma.h"
2
3 Caserma::Caserma() {
4     armiDepositate.inserisci("fucile di precisione", "arma da fuoco");
5     armiDepositate.inserisci("fucile", "fucile a canna liscia");
6     armiDepositate.inserisci("mitragliatrice", "arma da fuoco");
7 }
8
9 void Caserma::mostraArmi() {
10     armiDepositate.elenca_Armi();
11 }
12
13

```

file Caserma.cpp

```

1   #include "Armi.h"
2
3   Armi::Armi()
4   {
5       n_Armi= 0;
6   }
7
8   void Armi::inserisci(string n, string d)
9   {
10
11       pos = posizioneFisicaDaLogica(l_Armi, n_Armi );
12       l_Armi.inslista(Arma(n, d),pos);
13       n_Armi++;
14   }
15
16
17   int Armi::get_n_Armi()
18   {
19       return n_Armi;
20   }
21
22   string Armi::get_Arma(int i)
23   {
24       pos = posizioneFisicaDaLogica(l_Armi, i);
25       return l_Armi.legglistा(pos).stampa_Arma();
26   }
27
28   string Armi::get_nome_Arma(int i)
29   {
30       pos = posizioneFisicaDaLogica(l_Armi, i);
31       return l_Armi.legglistा(pos).get_nome();
32   }
33
34   string Armi::get_descrizione_Arma(int i)
35   {
36       pos = posizioneFisicaDaLogica(l_Armi, i);
37       return l_Armi.legglistा(pos).get_descrizione();
38   }
39
40   void Armi::elenca_Armi()
41   {
42       int i;
43
44       for (i = 0 ; i < get_n_Armi() ; i++)
45       {
46           cout << get_Arma(i);
47       }
48   }

```

file Armi.cpp

```

1  #include "Arma.h"
2  #include <iostream>
3
4  Arma::Arma()
5  {
6      descrizione="";
7      nome="";
8  }
9
10 Arma::Arma(string n, string d)
11 {
12     nome = n;
13     descrizione = d;
14 }
15
16 Arma::~Arma() {}
17
18 string Arma::get_descrizione() const
19 {
20     return descrizione;
21 }
22
23 void Arma::set_descrizione(string d)
24 {
25     descrizione = d;
26 }
27
28 string Arma::get_nome() const
29 {
30     return nome;
31 }
32
33 void Arma::set_nome(string n)
34 {
35     nome = n;
36 }
37
38 string Arma::stampa_Arma() const
39 {
40     return get_nome() + " - " + get_descrizione() + "\n";
41 }

```

file Arma.cpp

5.16.4 Luogo “caserma”

48	53,Nella caserma,000000030000,via 9,1,1
5	3,Nel corridoio,020446061932,via 1,3,3,via 3,1,1,via 9,1,1
126	53,3,via 9,est,1,1,1,1
127	3,53,via 9,ovest,1,1,1,1

Modifiche effettuata al file Mappa.h

```
64 |     typename Grafo<etichetta,peso>::nodo nodo_da_nome_etichetta(string);
```

Modifiche effettuate al file Mappa.cpp

```

2107 Grafo<Mappa::etichetta,Mappa::peso>::nodo Mappa::nodo_da_nome_etichetta(string pattern) //restituisce un nodo ricevendo in input (parte del) l'etichetta
2108 {
2109     Grafo<etichetta,peso>::nodo rif=*fittizio;
2110     int i=0; //-----+
2111     string testo; // +-- comodo
2112     int num=0; //-----+
2113     Lista<riferimento> risultati; //lista che conterrà i risultati
2114     Lista<riferimento>::posizione pos=risultati.primolista();
2115     bool trovato;
2116     string testo_low,pattern_low; //conterranno testo e pattern ma con tutti i caratteri minuscoli
2117     pattern_low=minuscolo(pattern); //conversione in minuscolo del pattern
2118     while (i<numnodi) //ricerca (CASE INSENSITIVE) in tutti i nodi
2119     {
2120         testo=riferimenti[i].leggietichetta().leggilocuogo(); //ricerca nel pattern nei nomi dei luoghi
2121         testo_low=minuscolo(testo); //conversione in minuscolo del testo
2122         trovato=multi_string_matching(testo_low,pattern_low); //string matching
2123         if (trovato) //se ci sono corrispondenze
2124         {
2125             risultati.inslista(riferimenti[i],pos); //aggiunge elemento in testa alla lista dei risultati
2126             num++;
2127         }
2128     else //se non ho trovato in nome
2129     {
2130         Lista<Vialuogo> vie=riferimenti[i].leggietichetta().leggivie(); //ricerca nel pattern in viali dei luoghi (analogia alla ricerca sopra)
2131         Lista<Vialuogo>::posizione ind=vie.primolista();
2132         while (!vie.finlista(ind) && !trovato)
2133         {
2134             testo=vie.leggilista(ind).legginome();
2135             testo_low=minuscolo(testo);
2136             trovato=multi_string_matching(testo_low,pattern_low);
2137             ind=vie.succlista(ind);
2138         }
2139         if (trovato)
2140         {
2141             risultati.inslista(riferimenti[i],pos);
2142             num++;
2143         }
2144     }
2145     i++;
2146 }
2147
2148 return(risultati.leggilista(risultati.primolista()).legginodo());
2149

```

5.17 RISTORANTE E FAST FOOD

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

Sono state effettuate diverse modifiche nei seguenti file:

- Astro.cpp
- Astro.h
- Lista.h
- Gioco.cpp
- Gioco.h
- Oggetto.cpp
- Oggetto.h
- Interfaccia.cpp
- Mappa.nav

5.17.1 Implementazione variabile sazietà

In Gioco.h andremo a dichiarare la variabile "sazio" di tipo intero, che rappresenterà il parametro indicante il livello di appetito del comandante. Essa sarà inizializzata, in Astro.cpp, a 120, in modo da permettere un normale svolgimento di gioco, e sarà decrementata ogni volta che verrà eseguita un'azione di un valore che varierà tra 1 e 4.

In Gioco.h:

```
int sazio;
```

In Astro.cpp:

```
sazio = 120;
```

5.17.2 Aggiornamento del dizionario

Verranno inseriti nel dizionario i nuovi termini che saranno utili al riconoscimento dei nuovi comandi.

5.17.2.1. Parole per il controllo della sazietà

```
vocabolario.inserisci("fame", 14);  
vocabolario.inserisci("appetito", 14);  
vocabolario.inserisci("sazio", 14);
```

5.17.2.2. Parole per l'acquisto delle pietanze

Sfrutteremo il termine "compra" già presente nel dizionario:

```
vocabolario.inserisci("compra", 33);  
vocabolario.inserisci("acquista", 33);
```

5.17.2.3. Aree Ristoro

```
vocabolario.inserisci("fast food", 15);  
vocabolario.inserisci("ristorante", 16);
```

5.17.2.4. Parole per controllare lo zaino e la tasca

```
vocabolario.inserisci("zaino", 28);  
vocabolario.inserisci("termico", 28);  
vocabolario.inserisci("tasca", 43);
```

5.17.2.5. Parole inerenti al buono pasto

```
vocabolario.inserisci("coupon", 44);  
vocabolario.inserisci("tesserino", 45);  
vocabolario.inserisci("buono", 59);  
vocabolario.inserisci("pasto", 59);
```

5.17.2.6. Parole inerenti agli alimenti delle zone ristoro

```

vocabolario.inserisci("trancio",46);
vocabolario.inserisci("pizza",46);
vocabolario.inserisci("hamburger",47);
vocabolario.inserisci("patatine",48);
vocabolario.inserisci("menu",49);
vocabolario.inserisci("completo",49);
vocabolario.inserisci("buono",59);
vocabolario.inserisci("pasto",59);
vocabolario.inserisci("spinaci",92);
vocabolario.inserisci("arrosto",93);
vocabolario.inserisci("filetto",94);
vocabolario.inserisci("frutti",95);
vocabolario.inserisci("mare",95);
vocabolario.inserisci("fettuccine",96);
vocabolario.inserisci("paglia",96);
vocabolario.inserisci("fieno",96);
vocabolario.inserisci("gelato",97);
vocabolario.inserisci("mangia",98);

```

5.17.3 Aggiornamento del dizionario degli oggetti

Questi oggetti saranno inseriti nell'apposito dizionario.

5.17.3.1. Oggetti "zaino termico" e "tasca"

```

oggetti.inserisci(Oggetto("uno zaino termico",28,6));
oggetti.inserisci(Oggetto("tasca", 43,0));

```

5.17.3.2. Inserimento degli alimenti nel dizionario degli oggetti

```

oggetti.inserisci(Oggetto("Hamburger",47,22,17,"BLU"));
oggetti.inserisci(Oggetto("Trancio di pizza",46,22,20,"BLU"));
oggetti.inserisci(Oggetto("Patatine fritte",48,22,14,"BLU"));
oggetti.inserisci(Oggetto("Menu completo",49,22,45,"VERDE"));
oggetti.inserisci(Oggetto("Gelato alla panna",97,22,24,"GIALLO"));
oggetti.inserisci(Oggetto("Spinaci alla Panna",92,21,28,"GIALLO"));
oggetti.inserisci(Oggetto("Arrosto",93,21,47,"VERDE"));
oggetti.inserisci(Oggetto("Filetto",94,21,33,"GIALLO"));
oggetti.inserisci(Oggetto("Fettuccine Paglia e
fieno",96,19,46,"VERDE"));
oggetti.inserisci(Oggetto("Misto frutti di mare",95,21,38,"VERDE"));
oggetti.inserisci(Oggetto("un tesserino ristorante
verde",45,1,"VERDE"));
oggetti.inserisci(Oggetto("un buono pasto giallo",59,5,"GIALLO"));
oggetti.inserisci(Oggetto("un coupon fast food blu",44,6,"BLU"));
oggetti.inserisci(Oggetto("un tesserino ristorante
verde",45,1,"VERDE"));

```

5.17.3.3. Inserimento dei buoni pasto nel dizionario degli oggetti

Come detto in precedenza, i buoni pasto saranno di diverse categorie:

```
oggetti.inserisci(Oggetto("un tesserino ristorante  
verde",45,1,"VERDE"));  
oggetti.inserisci(Oggetto("un buono pasto giallo",59,5,"GIALLO"));  
oggetti.inserisci(Oggetto("un coupon fast food blu",44,6,"BLU"));
```

5.17.4 Aggiunta delle azioni di gioco

Breve spiegazione sulla codifica dei comandi.

I comandi che andremo ad inserire verranno codificati tramite l'espressione:

[Codifica comando = LL *10000+VV*100+OO]

LL = Codice del luogo il cui comando può essere impartito.

VV = Codice del verbo del dizionario che deve essere presente necessariamente, valido ad indicare una certa azione.

OO = Codice indicante l'oggetto.

Di seguito vedremo come verranno implementate nel file Astro.cpp.

5.17.4.1. Azione per il controllo della sazietà

Questa azione ci permetterà di controllare il livello di sazietà del comandante, mediante l'utilizzo dei vocaboli "sazio", "fame" e "appetito".

```
azioni.inserisci(1400, 75);
```

5.17.4.2. Azioni per l'acquisto di cibo nel fast food e nel ristorante

Queste azioni ci permetteranno di comprare del cibo presso il fast food e il ristorante tramite il vocabolo "compra" o "acquista".

```
azioni.inserisci(223399, 76); //compra gli oggetti nel fast food  
azioni.inserisci(213399, 76); //compra gli oggetti nel ristorante
```

5.17.4.3. Azioni per la consumazione del cibo presenti nello zaino termico

Utilizzeremo queste azioni per consumare del cibo presente nello zaino, per mezzo del vocabolo "mangia" e indicando uno dei piatti a disposizione.

```
azioni.inserisci(229846, 77); //mangia i piatti conservati nello zaino
```

```
azioni.inserisci(229847, 77); //codici dei piatti
```

```

azioni.inserisci(229848, 77);
azioni.inserisci(229849, 77);
azioni.inserisci(219892, 77);
azioni.inserisci(219893, 77);
azioni.inserisci(219894, 77);
azioni.inserisci(219895, 77);
azioni.inserisci(219896, 77);
azioni.inserisci(219897, 77);

```

5.17.4.4. Azioni per la consumazione della tasca e dello zaino

Per mezzo di queste azioni, richiamate con i vocaboli "zaino", "termico" (o insieme) potremo controllare il cibo posseduto. Mentre con il vocabolo "tasca" potremo controllare di quali buoni pasto disporremo.

```

azioni.inserisci(2800, 79); //mostra il contenuto dello zaino termico
azioni.inserisci(4300, 78); //mostra il contenuto della tasca

```

5.17.5 Classe Oggetto: Modifiche da effettuare

Breve spiegazione delle modifiche da apportare nella classe oggetto.

Modificheremo la classe oggetto in modo da poterla utilizzare anche con i buoni pasto e con i cibi. Utilizzeremo due nuovi costruttori, uno per i buoni pasto e l'altro per il cibo.

5.17.5.1. Costruttore per i buoni pasto

Inseriremo un nuovo costruttore, dedicato agli oggetti "Buoni Pasto", strutturato nel seguente modo:

```

Oggetto::Oggetto($STRINGA n, INTERO c, INTERO i, INTERO cat){
    nome ← n;
    codice ← c;
    luogo ← i;
    category ← cat;
}

```

In oggetto.h inseriremo:

```
Oggetto(string n,int c,int l,string cat);
```

In oggetto.cpp aggiungeremo:

```
Oggetto::Oggetto(string n, int c, int l, string cat) {
    nome = n;
    codice = c;
    luogo = l;
    category = cat;
}
```

5.17.5.2. Costruttore per i cibi

Il costruttore per i cibi sarà strutturato nel seguente modo:

```
Oggetto::Oggetto(STRINGA n, INTERO c, INTERO l, INTERO v, STRINGA cat) {
    nome ← n;
    codice ← c;
    luogo ← l;
    valore_e ← v;
    category ← cat;
}
```

Inseriremo nel file Oggetto.h:

```
Oggetto(string n, int c, int l, int v, string cat);
```

Mentre nel file Oggetto.cpp:

```
Oggetto::Oggetto(string n, int c, int l, int v, string cat) {
    nome = n;
    codice = c;
    luogo = l;
    valore_e = v;
    category = cat;
}
```

5.17.5.3. Implementazione del metodo get_categoria

Il metodo get_categoria, sarà realizzato come segue, utile per confrontare le categorie di oggetti nel momento in cui si effettua l'azione compra:

```
STRINGA Oggetto::get_categoria()
INIZIO
    return category;
FINE
```

In oggetto.h inseriremo quindi:

- Nella sezione **public**: `string get_categoria();`
- Nella sezione **private**: `int valore_e;`

Mentre in oggetto.cpp:

```
string Oggetto::get_categoria()  
{  
    return category;  
}
```

5.17.5.4. Implementazione del metodo get_valore_e

Per ottenere energia dal cibo, realizzeremo il metodo get_valore_e nel seguente modo:

```
INTERO Oggetto::get_valore_e()  
INIZIO  
    RESTITUISCI valore_e;  
FINE
```

Aggiungeremo nel file Oggetto.h:

- Nella sezione public: int get_valore_e();
- Nella sezione private: string category;

Mentre nel file Oggetto.cpp:

```
string Oggetto::get_categoria()  
{  
    return category;  
}
```

5.17.6 File Mappa.nav: Modifiche effettuate

Per creare le zone ristoro, verrà modificato il file mappa.nav in modo da inserire due nuove stanze all'interno dell'astronave. In questo file vi sono diverse informazioni, ovvero:

- Numero complessivo delle stanze visitabili durante il gioco;
- Codice e nome di ogni stanza;
- Le direzioni in cui è possibile spostarsi da ogni stanza;
- Le istruzioni per il navigatore;
- I cammini che rendono possibili i collegamenti da una stanza ad un'altra, utili al navigatore.

Verrà aumentato il numero delle stanze visitabili, da 20 a 22. Gli ID del ristorante e del fast food saranno 21 e 22 rispettivamente.

Verrà aggiunto al file mappa.nav:

```
[...]
21,Nel Ristorante,000000000006, via 3,1,1
22,Nel Fast Food,000000000600, via 3,6,6 [...]

6,21,via 3,sali,4,4,1,1
21,6,via 3,scendi,4,4,1,1
6,22,via 3,scendi,4,4,1,1
22,6,via 3,sali,4,4,1,1
```

Il luogo 6 (La cabina) verrà modificato come segue:

```
6,Nella tua cabina,000003002122,via 3,2,2
```

Verrà anche modificato il numero dei luoghi della mappa da 20 a 22.

5.17.7 Procedure

Di seguito saranno specificate le procedure da implementare per rendere efficaci le integrazioni delle nuove funzionalità.

5.17.7.1. Monitoraggio della sazietà

Questo metodo verrà implementato nel file Astro.cpp aggiungendo l'azione_75, permettendoci di visualizzare il livello di sazietà tramite l'apposito comando.

Integrazione nel metodo `bool Astro::esegui_specifiche(int a, Mappa &M)`:

```
case 75:
    azione_75();
    break;
```

Implementazione della procedura azione_75 in Astro.cpp:

```
//Azione fame: monitora l'appetito del comandante, funzione richiamabile in qualsiasi luogo
void Astro::azione_75() //{
{
    string messaggio;
    string stato;
    if (sazio <= 120 && sazio >= 80){
        stato = "OTTIMALE";
        messaggio = "Inizio a sentire un leggero languorino...";
    }
    else if (sazio <= 79 && sazio >= 50){
        messaggio = "E' tutto il giorno che non mangio, dovrei fare una pausa...";
        stato = "BUONO";
    }
    else if (sazio <= 49 && sazio >= 30){
        messaggio = "Accidenti, la fame aumenta. Mi servono energie, e subito!";
        stato = "PESANTE";
    }
}
```

```

        stato = "CRITICO";
    }
    else if (sazio <= 29 && sazio >=0) {
        messaggio = "...Non riesco a fare un altro passo... le forze mi abbandonano...";
        stato = "MOLTO CRITICO";
    }
    cout << "\nlivello sazieta': " << stato << " (" << sazio << ")" << endl;
    interfaccia.scrivi(messaggio);

}

```

5.17.7.2. Gestione del tempo rimanente

La funzionalità della sazietà del giocatore, come detto prima, influenzerà il tempo a disposizione per concludere l'avventura. Realizzeremo ciò, grazie all'integrazione della procedura aggiorna_tempo nel file Gioco.cpp, che sarà implementata nel seguente modo:

```

// Integrazione fatta alla funzione che mi permette di gestire il tempo anche in base alla sazietà
del //personaggio

srand(time(NULL));
int x = rand()%4+1;
if (sazio <= 120 && sazio >= 80)
{
    tempo--;
    if (sazio <= 90 && sazio >= 80)
        interfaccia.scrivi("\nInizio a sentire un leggero languorino...");

}
else if (sazio <= 79 && sazio >= 50)
{
    tempo = tempo - 2;
    if (sazio <= 60 && sazio >= 50)
        interfaccia.scrivi("\nE' tutto il giorno che non mangio, dovrei fare una
pausa...");

}
else if (sazio <= 49 && sazio >= 20)
{
    tempo = tempo - 3;
    if (sazio <= 30 && sazio >= 20)
        interfaccia.scrivi("\nAccidenti, la fame aumenta. Mi servono energie, e
subito!");

}

else if (sazio <= 19 && sazio >=0)
{
    tempo = tempo - 4;
    if (sazio <= 10 && sazio >= 0)
        interfaccia.scrivi("\n...Non riesco a fare un altro passo... le forze mi
abbandonano...");
}

```

```

        }
sazio = sazio - x;

```

5.17.7.3. Gestione degli acquisti del cibo

Con questo metodo potremo gestire gli acquisti riguardante il cibo che, una volta acquistato, finirà nello zaino, pronto per essere consumato all'occorrenza. Inoltre utilizzeremo un controllo per verificare se il giocatore possiede lo zaino e/o i buoni pasto necessari per comprare il cibo.

Il seguente metodo sarà implementato nel file Astro.cpp, aggiungendo nel file originale una nuova azione, chiamata `azione_76`.

In `bool Astro::esegui_specifiche (int a, Mappa &M)` implementeremo:

```

case 76:
    azione_76();
    break;

```

Implementazione della procedura `azione_76` in Astro.cpp:

```

void Astro::azione_76() // 
{
    riferimento_zaino=zaino_frigo.primolista();
    while(!zaino_frigo.finelista(riferimento_zaino))
    {
        riferimento_zaino=zaino_frigo.succlista(riferimento_zaino);
    }

    if(oggetti.get Oggetto(55).get_luogo()==0)
    {
        riferimento_tasca=tasca.primolista();
        bool trovato=false;
        if(tasca.listavuota())
            interfaccia.scrivi("Non hai Buoni Mensa! Procurateli per poterli scambiare con deliziose
pietanze!");
        else
        {
            while(!tasca.finelista(riferimento_tasca) && !trovato)
            {

                if(oggetti.get Oggetto(tasca.legglista(riferimento_tasca)).get_categoria()==oggetti.get Oggetto(og)
                .get_categoria())
                {
                    oggetti.set_luogo(tasca.legglista(riferimento_tasca),-99);
                    tasca.canlista(riferimento_tasca);
                }
            }
        }
    }
}

```

```

        oggetti.set_luogo(og,0);
        interfaccia.scrivi("Piatto acquistato ed inserito nello zaino termico!");
        zaino_frigo.inslista(riferimento_zaino, og);
        trovato=true;
    }
    else
        riferimento_tasca=tasca.succlista(riferimento_tasca);
    }
    if(!trovato)
    {
        interfaccia.scrivi("Non possiedi un Buono Mensa della stessa categoria del piatto scelto.");
    }
}
else
    interfaccia.scrivi("Non hai nessuno zaino termico! Dovresti prima prenderlo.");
}

```

5.17.7.4. Miglioramento del livello energetico

Il metodo che verrà mostrato servirà ad aggiungere alla variabile rappresentante la sazietà del comandante, l'apporto calorico del cibo che si vorrà mangiare. Verrà implementata una nuova azione nel file Astro.cpp (azione_77).

In bool Astro::esegui_specifiche (int a, Mappa &M) implementeremo:

```

case 77:
    azione_77();
    break;

```

Implementazione della procedura azione_77 in Astro.cpp:

```

//Azione mangia: Mangia i piatti che ha comprato
void Astro::azione_77() // 
{
    string messaggio;
    int y;
    srand(time(NULL));
    y= rand()%3+1;
    switch (y)
    {
        case 1:
            messaggio = "Che bonta'! Ci voleva proprio!";
            break;
        case 2:
            messaggio = "Mai assaggiato un piatto cosi' buono!";
            break;
    }
}
```

```

        case 3:
            messaggio = "Era ora di mettere qualcosa sotto i denti!";
            break;
    }
    bool trovato = false;
    riferimento_zaino = zaino_frigo.primolista();

    while(!zaino_frigo.finelista(riferimento_zaino) && trovato == false)
    {
        if (oggetti.get_oggetto(zaino_frigo.legglista(riferimento_zaino)).get_codice() == oggetti.get_oggetto(og).get_codice())
        {
            sazio = sazio + oggetti.get_oggetto(og).get_valore_e();
            oggetti.set_luogo(og, -99);
            zaino_frigo.canclista(riferimento_zaino);
            trovato = true;
            cout << "\n" << messaggio << "\n";
        }
        else
            riferimento_zaino = zaino_frigo.succlista(riferimento_zaino);
    }
    if (trovato == false)
        interfaccia.scrivi("Non hai a disposizione quel piatto!");
}

```

5.17.7.5. Visualizzazione del contenuto della tasca

Questo metodo ci permetterà di visualizzare la presenza di buoni pasto nella tasca del comandante, permettendo al giocatore di decidere al meglio come spenderli, oppure spingerlo a cercarne degli altri. Sarà implementata una nuova azione (azione_78) nel file Astro.cpp.

In `bool Astro::esegui_specifiche (int a, Mappa &M)` implementeremo:

```

case 78:
    azione_78();
    break;

```

L' implementazione della procedura `azione_78()` in Astro.cpp sarà:

```

//Azione Tasca: mostra il contenuto della tasca del capitano
void Astro::azione_78() // 
{
    riferimento_tasca=tasca.primolista();
    if(tasca.listavuota())
        interfaccia.scrivi("Non hai nulla in tasca.");
    else
    {
        interfaccia.scrivi("In tasca hai:");
        do

```

```

    {
        if (tasca.finelista(riferimento_tasca))
            cout << "\n- " << oggetti.get_oggetto(tasca.legglista(riferimento_tasca)).get_nome()
            << ". ";
        else
            cout << "\n- " << oggetti.get_oggetto(tasca.legglista(riferimento_tasca)).get_nome()
            << ";" ;
            riferimento_tasca=tasca.succlista(riferimento_tasca);
    }
    while(!tasca.finelista(riferimento_tasca));
    cout<<endl;
}
}

//Azione Zaino Termico: mostra il contenuto dello Zaino Termico

```

5.17.7.6. Visualizzazione del contenuto dello zaino

Questo metodo permetterà al giocatore di visualizzare il contenuto presente nello zaino, ovvero il cibo acquistato di cui dispone pronto per essere consumato. Per realizzarlo, Implementeremo una nuova azione (azione_79) nel file Astro.cpp

In `bool Astro::esegui_specifiche (int a, Mappa &M)` implementeremo:

```

case 79:
    azione_79();
    break;

```

L' implementazione della procedura `azione_79()` in Astro.cpp sarà:

```

//Azione Zaino Termico: mostra il contenuto dello Zaino Termico
void Astro::azione_79() // 
{
    if(oggetti.get_oggetto(65).get_luogo()==0)
    {
        riferimento_zaino=zaino_frigo.primolista();
        if(zaino_frigo.listavuota())
            interfaccia.scrivi("Non c'e' nulla nello zaino termico. Qui puoi conservare il
cibo!");
        else
        {
            interfaccia.scrivi("Nello zaino termico hai messo:");
            do
            {
                if (zaino_frigo.finelista(riferimento_zaino))
                    cout << "\n- " <<
oggetti.get_oggetto(zaino_frigo.legglista(riferimento_zaino)).get_nome() << ".";
                else

```

```

        cout << "\n- " <<
oggetti.get Oggetto(zaino_frigo.leggilista(riferimento_zaino)).get Nome() << ";" ;
riferimento_zaino=zaino_frigo.succlista(riferimento_zaino);
}
while(!zaino_frigo.finelista(riferimento_zaino));
cout<<endl;
}
}
else
interfaccia.scrivi("Non hai nessuno zaino termico! Dovresti prima prenderlo.");
}

```

5.17.8 Altre modifiche

Parleremo adesso di alcune modifiche particolari che andremo ad effettuare all'interno del codice per rendere funzionali i metodi citati in precedenza.

5.17.8.1. Modifica del metodo lascia

Il metodo lascia sarà modificato in modo da adattarlo agli oggetti contenuti nello zaino e/o nella tasca.

```

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.finelista(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.finelista(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);
}

oggetti.set_luogo(og,luogo_attuale);
interfaccia.scrivi("Fatto.");
}
}
else
if (og == 67)
    interfaccia.scrivi("Non puoi compiere questa azione!");
else
    interfaccia.scrivi("Non puoi lasciare oggetti in questa stanza.");
}

}

```

5.17.8.2. Modifiche al metodo init

In Gioco.cpp nel metodo `void Gioco::init()` verrà effettuata la seguente aggiunta:

```

tasca.svuota();
zaino_frigo.svuota();

```

5.17.8.3. Modifiche al metodo prendi

In Gioco.cpp il metodo prendi verrà completamente riscritto, in quanto non funzionante e non adatto. Sarà realizzato in modo da mantenere le funzionalità di base. In sostanza il metodo prendi funzionerà sia per il portafoglio, che per la tasca.

Di seguito l'implementazione del metodo prendi in Gioco.cpp:

```

void Gioco::prendi()
{
    //modifica Gallone 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;

    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);
    }

    if (oggetti.get_oggetto(og).get_luogo() == 0)
        interfaccia.scrivi("- Gia' fatto.");
    else if (oggetti.get_oggetto(og).get_luogo() < 0)
        interfaccia.scrivi("- Non e' possibile.");

    else if ((!prendi_specifiche() && (og == 27)) || !prendi_specifiche() && (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");
        }
    }
}

// Metodo prendi per lo zaino e per la tasca .
else if (!prendi_specifiche())
{
    if (og >=77 && og <= 79)
    {

        oggetti.set_luogo(og, 0);
        tasca.inslista(riferimento_tasca,og);
        interfaccia.scrivi("Inserito nella tasca!");

    }
    else if (og >= 57 && og <=66)
    {

        if (tasca.listavuota()) {
            interfaccia.scrivi("Non hai buoni pasto per acquistarli!");
    }
}

```

```

        cpz=false;}
    else{interfaccia.scrivi("Per ottenere le pietanze dovresti comprarle usando
i buoni pasto.");
        cpz=false; }

}

else oggetti.set_luogo(og,0);

//Fine Modifica Gallone
if (!preso_specifiche() && ( cpz))
    interfaccia.scrivi("Fatto.");
}

}

bacheca.CancellaMessaggioGioco("*non uscire fuori dall'astronave se non hai
l'equipaggiamento da astronauta");
}

```

E' stata implementata una funzione che bloccherà il giocatore in caso di utilizzo del vocabolo "prendi" anziché "compra" sul cibo.

```

else if (og >= 67 && og <=76)

{
    if (tasca.listavuota()) {
        interfaccia.scrivi("Non hai buoni pasto per acquistarlo!");
        cpz=false;
    }
    else{interfaccia.scrivi("Per ottenere le pietanze dovresti comprarle
usando i buoni pasto.");
        cpz=false; }

}

```

5.17.8.4. Dichiaraione delle nuove procedure

Le seguenti dichiarazioni saranno aggiunte in Astro.h:

```

void azione_75(); //fame-appetito-sazio
void azione_76(); //acquista-compra
void azione_77(); //mangia
void azione_78(); //controlla contenuto tasca
void azione_79(); //controlla contenuto dello zaino termico

```

5.17.8.5. Modifica del metodo elenca_oggetti

In Interfaccia.cpp sarà modificato il metodo elenca_oggetti in modo da visualizzare l'inventario e gli oggetti contenuti nella tasca e nello zaino, ovvero cibo e buoni pasto.

```
void Interfaccia::elenca_oggetti(Oggetti oggetti, string premessa)
{
    int n_oggetti = oggetti.get_n_oggetti();

    if (n_oggetti > 0)
    {
        cout << premessa;

        //Inizio modifica Gallone - Modifico il metodo in modo che mi faccia vedere i
        differenti attributi di ogni oggetto
        // in base al particolare oggetto
        for (int i = 1; i <= n_oggetti; i++)
        {
            if (oggetti.get_oggetto(i).get_luogo() == 21 || oggetti.get_oggetto(i).get_luogo() ==
22)
            {
                if (i == n_oggetti)

                    cout << "\n- " << oggetti.get_oggetto(i).get_nome() << ", Valore energetico: " <<
oggetti.get_oggetto(i).get_valore_e() << ", Categoria: " << oggetti.get_oggetto(i).get_categoria() <<
".";
                else

                    cout << "\n- " << oggetti.get_oggetto(i).get_nome() << ", Valore energetico: " <<
oggetti.get_oggetto(i).get_valore_e() << ", Categoria: " <<
oggetti.get_oggetto(i).get_categoria() << ";" ;
            }
            else if (oggetti.get_oggetto(i).get_luogo() == 0)

            {
                if ((oggetti.get_oggetto(i).get_codice() < 80 ||

oggetti.get_oggetto(i).get_codice() > 94)) //da modificare in base al numero di oggetti

                {
                    if (i==n_oggetti)

                        cout << "\n- " << oggetti.get_oggetto(i).get_nome() << ".";
                    else

                        cout << "\n- " << oggetti.get_oggetto(i).get_nome() << ";" ;
                }
            }
            else

            {
                if (i == n_oggetti)

                    cout << "\n- " << oggetti.get_oggetto(i).get_nome() << ".";
                else

                    cout << "\n- " << oggetti.get_oggetto(i).get_nome() << ";" ;
            }
        }
    }
    //Fine Gianmarco Gallone
    cout << endl;
}
```

```
}
```

5.18 ESTRAZIONE DI UN PERSONAGGIO CASUALE

Ambiente di sviluppo: Code Blocks

Nel realizzare le modifiche richieste è stata aggiunta la classe Personaggi.h e Personaggi.cpp e la struttura dati Insieme.h

Sono stati modificati i seguenti file: astro.h, astro.cpp, gioco.h, gioco.cpp e lista.h

Nel codice, le modifiche sono indicate tramite commenti del tipo:

```
// inizio modifiche ZAGARIA
```

```
.
```

```
.
```

```
// fine modifiche ZAGARIA
```

Per prima cosa, è stata aggiunta la classe personaggi:

- Personaggi.h

```
/*
 * Personaggi.h

*
* Created on: 13/giu/2015
* Author: marta
*/
```

```
#ifndef PERSONAGGI_H_
```

```

#define PERSONAGGI_H_


#include<iostream>
#include<cstring>
#include<stdarg.h>

using namespace std;

class Personaggi
{
public:
    Personaggi();
    ~Personaggi();
    //Setter
    void setNome(string);
    void setFrasì(string, string);
    void setLuogo(int);
    void setOgg(bool, int, ...);
    void setFrasiconOgg1(string, string);
    void setFrasiconOgg2(string, string);
    //Getter
    string getNome();
    int getLuogo();
    bool getIfNec();
    int get_ogg1();
    int get_ogg2();
    //Metodi
    void SelezionaFrase();
    void SelezionaFrase1();
    void SelezionaFrase2();
    void stampa();

private:
    string nome;

```

```

        stringfraseAiuto;
        stringfraseAiutoconOgg1;
        stringfraseAiutoconOgg2;
        stringfraseIntralcio;
        stringfraseIntralcioconOgg1;
        stringfraseIntralcioconOgg2;
        intluogo;
        intogg1,ogg2;
        boologg_necessario;
    };

```

```
#endif/* PERSONAGGI_H */
```

- e Personaggi.cpp...

```

#include"Personaggi.h"
#include<iostream>
#include<cstdlib>
#include<ctime>
#include<stdarg.h>

usingnamespace std;

Personaggi::Personaggi() {}

Personaggi::~Personaggi() {}

voidPersonaggi::setNome( string a)
{
    nome=a;
    ogg_necessario=false;
}

```

```

}

void Personaggi::setFrasi(string a, string b)
{
    fraseAiuto=a;
    fraseIntralcio=b;
}

void Personaggi::setLuogo(int a)
{
    luogo=a;
}

string Personaggi::getNome()
{
    return nome;
}

int Personaggi::getLuogo()
{
    return luogo;
}

void Personaggi::SelezionaFrase()
{
    int v;
    v= rand() % 2;
    if (v==1)
    {
        cout<<fraseAiuto<<endl;
    }
    elseif (v==0)
    {
        cout<<fraseIntralcio<<endl;
    }
}

```

```

        }

    else

    {

        cout<<"error"<<endl;
    }

}

voidPersonaggi::stampa()

{

    cout<<nome<<"ti dice"<<endl;
    SelezionaFrase();
}

voidPersonaggi::setOgg(bool a ,int b, ...)

{

    va_list marcatore;
    va_start(marcatore,b);
    ogg_necessario=a;
    ogg1=b;
    if(va_arg(marcatore,int)==NULL)
    {
        ogg2=0;
    } else
    {
        ogg2=va_arg(marcatore,int);
    }
    va_end(marcatore);
}

boolPersonaggi::getIfNec()

{

    returnogg_necessario;
}

intPersonaggi::get_ogg1()

```

```

{
    returnogg1;
}

intPersonaggi::get_ogg2()
{
    returnogg2;
}

voidPersonaggi::setFrasiconOgg1(string a, string b)
{
    fraseAiutoconOgg1=a;
    fraseIntralocioconOgg1=b;
}

voidPersonaggi::setFrasiconOgg2(string a, string b)
{
    fraseAiutoconOgg2=a;
    fraseIntralocioconOgg2=b;
}

voidPersonaggi::SelezionaFrase1()
{
    int v;
    v= rand() % 2;
    if(v==1)
    {
        cout<<nome<<"ti dice"<<endl;
        cout<<fraseAiutoconOgg1<<endl;
        cout<<endl;
    }
    elseif(v==0)
    {
        cout<<nome<<"ti dice"<<endl;
        cout<<fraseIntralocioconOgg1<<endl;
    }
}

```

```

        }

    else

    {

        cout<<"error"<<endl;
    }

}

voidPersonaggi::SelezionaFrase2()

{

    int v;

    v= rand() % 2;

    if(v==1)

    {

        cout<<nome<<"ti dice"<<endl;

        cout<<fraseAiutoconOgg2<<endl;

        cout<<endl;

    }

    elseif(v==0)

    {

        cout<<nome<<"ti dice"<<endl;

        cout<<fraseIntralcioconOgg2<<endl;

    }

    else

    {

        cout<<"error"<<endl;
    }

}

```

- Aggiunta la struttura dati Insieme.h per la gestione dei personaggi

```

#ifndef INSIEME_H_
#define INSIEME_H_
#include"Lista.h"
#include<iostream>

```

```

#include<cstdlib>
#include<ctime>
usingnamespace std;

template<class tipoelem>
class Insieme
{
public:
    typedef typename Lista<tipoelem>::posizionepos;

    Insieme();
    ~Insieme();
    void creainsieme();
    bool insiemevuoto();
    bool appartiene(tipoelem);
    bool esiste(int);
    void inserisci(tipoelem);
    void cancella(tipoelem);
    void unione(Insieme*, Insieme*, Insieme*);
    void intersezione(Insieme*, Insieme*, Insieme*);
    void differenza(Insieme*, Insieme*, Insieme*);
    int cardinalita();
    tipoelem getElem(int);

    //OPERATORI DI SERVIZIO
    void stampa(Insieme*);
    void distruggi(Insieme*);

private:
    Lista<tipoelem> setlist;
    int n_elementi;
};

```

```

template<class<tipoelem>
Insieme<tipoelem>::Insieme()
{
    creainsieme();
    n_elementi = 0;
}

template<class<tipoelem>
Insieme<tipoelem>::~Insieme() { }

template<class<tipoelem>
void Insieme<tipoelem>::creainsieme()
{
    setlist.crealista();
}

template<class<tipoelem>
bool Insieme<tipoelem>::insiemevuoto()
{
    if (setlist.listavuota())
        return true;
    else
        return false;
}

template<class<tipoelem>
bool Insieme<tipoelem>::appartiene(tipoelem ele)
{
    pos p;
    bool trovato = false;
    p = setlist.primolista();
    while (!setlist.finelista(p) && !trovato) {
        if (setlist.leggilista(p).getNome() == ele.getNome())
            trovato = true;
    }
}

```

```

        else
            p = setlist.succlista(p);
    }

    return trovato;
}

template<classtipoelem>
void Insieme<tipoelem>::inserisci(tipoelem ele)
{
    pos p;
    if (!appartiene(ele))
    {
        p = setlist.primolista();
        setlist.inslista(p,ele);
        ++(n_elementi);
    }
}

template<classtipoelem>
void Insieme<tipoelem>::cancella(tipoelem ele)
{
    bool trovato = false;
    pos p;
    p = setlist.primolista();
    while (!setlist.finelista(p) && !trovato){
        if(setlist.legglista(p).getNome() == ele.getNome())
            trovato = true;
        else
            p = setlist.succlista(p);
    }
    if (setlist.legglista(p).getNome() == ele.getNome())
        setlist.canclista(p);
}

```

template<classtipoelem>

```

voidInsieme<tipoelem>::unione(Insieme* A, Insieme* B, Insieme* C)
{
    // Oppure si può usare l'operatore INSERISCI dell'insieme

    pos p;

    p = A->setlist.primolista();
    while (!A->setlist.finelista(p))
    {
        C->setlist.inslista(A->setlist.legglista(p), C-
        >setlist.primolista());
        p = A->setlist.succlista(p);
    }

    p = B->setlist.primolista();
    while (!B->setlist.finelista(p))
    {
        C->setlist.inslista(B->setlist.legglista(p), C-
        >setlist.primolista());
        p = B->setlist.succlista(p);
    }

    C->setlist.epurazione(&C->setlist);
}

template<class tipoelem>
voidInsieme<tipoelem>::intersezione(Insieme* A, Insieme* B, Insieme* C)
{
    pos p, q;
    p = A->setlist.primolista();
    while (!A->setlist.finelista(p))
    {
        q = B->setlist.primolista();
        while (!B->setlist.finelista(q))
        {
            if (A->setlist.legglista(p) == B->setlist.legglista(q))
}
}

```

```

        {
            C->inserisci(B->setlist.legglista(q));
        }
        q = B->setlist.succlista(q);
    }
    p = A->setlist.succlista(p);
}
}

template<class tipoelem>
void Insieme<tipoelem>::differenza( Insieme* A, Insieme* B, Insieme* C)
{
    pos p;
    p = A->setlist.primolista();
    while (!A->setlist.finelista(p))
    {
        C->inserisci(A->setlist.legglista(p));
        p = A->setlist.succlista(p);
    }

    p = B->setlist.primolista();
    while (!B->setlist.finelista(p))
    {
        if (C->appartiene(B->setlist.legglista(p)))
            C->cancella(B->setlist.legglista(p));
        p = B->setlist.succlista(p);
    }
}

template<class tipoelem>
void Insieme<tipoelem>::stampa( Insieme* A)
{
    int x = 1;
    pos p;
    p = A->setlist.primolista();
}

```

```

while (!A->setlist.finelista (p) )
{
    setlist.legglista (p).stampa();
    cout<<"\n";
    x++;
    p = A->setlist.succlista (p);
}
}

template<classtipoelem>
void Insieme<tipoelem>::distruggi (Insieme* A)
{
    while (!A->setlist.finelista (A->setlist.primolista()))
        A->setlist.canclista (A->setlist.primolista());
}

template<classtipoelem>
tipoelem Insieme<tipoelem>::getElem (int luogo)
{
    pos p;
    tipoelem elem;
    pos tro[10];
    int i =0;
    p=setlist.primolista();
    while (!setlist.finelista (p)) {
        if (luogo==setlist.legglista (p).getLuogo ())
        {
            tro[i]=p;
            i++;
            p=setlist.succlista (p);
        }
        else
            p=setlist.succlista (p);
    }
    int v= rand() % i;
}

```

```

        elem=setlist.legglista(tro[v]);
        cancella(elem);
        n_elementi--;
        return elem;
    }

template<classestipoelem>
bool Insieme<tipoelem>::esiste(int luogo)
{
    pos p;
    bool trovato = false;
    p=setlist.primolista();
    while (!setlist.finlista(p)) {
        if(luogo==setlist.legglista(p).getLuogo()) {
            trovato=true;
            p=setlist.succlista(p);
        }
        else
            p=setlist.succlista(p);
    }
    return trovato;
}
#endif/* INSIEME_H_ */

```

- Astro.h

Aggiunta inizializzazione lista oggetti

```
// iniziomodificheZagaria
```

```
Lista<Oggetti>lisOggetti;
```

```
// fine modificheZagaria
```

- Astro.cpp

Inserimento di tutte le informazioni legate ai personaggi: in particolare nome del personaggio, e relativo codice che, come detto in precedenza ha la funzione di costruire il comando “ Interagisci con Cloud” (esempio)

```
//IniziomodificheZagaria
```

```
vocabolario.inserisci("cloud", 73);
vocabolario.inserisci("cid", 73);
vocabolario.inserisci("vincent", 73);
vocabolario.inserisci("kain", 73);
vocabolario.inserisci("cecil", 73);
vocabolario.inserisci("garnet", 73);

.
.

vocabolario.inserisci("dom", 73);
```

Aggiunte le azioni comunica e interagisci che servono al protagonista per relazionarsi con il personaggio, anche qui con il rispettivo codice

```
vocabolario.inserisci("interagisci", 39);
vocabolario.inserisci("comunica", 39);
vocabolario.inserisci("con", 7);
```

Comando che mi permette di costruire la frase di interazione (es: “Comunica con Garnet”)

```
//IniziomodificheZagaria
```

```

azioni.inserisci(3973,43);

//fine modifcheZagaria

tempo = 350; //tempopassato a 350 anzichè 300-->zagaria

```

- Gioco.h

Inserito l' include dei nuovi file aggiunti al progetto Personaggi.h e Insieme.h

```

//iniziomodifcheZagaria-- aggiuntadelleclassiPersonaggi e Insieme --
#include "Personaggi.h"
#include "Insieme.h"

// fine modifcheZagaria

```

Inizializzazione della nuova parola2

```

// iniziomodificaZagaria

stringparola2;

// fine modifcheZagaria

```

Inserimento dell' insieme personaggi e alcune inizializzazioni

```

// iniziomodifica ZAGARIA -- inserimento dell'insiemepersonaggi

Insieme<Personaggi>insPersonaggi;

```

```

Personaggipg;
intluogo_precedente;
boolpresente2;
intindex;

// fine modifica ZAGARIA

```

Aggiunte le relative funzioni inerenti ai personaggi, ad esempio la funzione parla che permette al protagonista di parlare con il personaggio, l' entrata e l' uscita del personaggio, ecc..

```

// iniziomodifiche ZAGARIA-- aggiunteledichiarazionidifunzioni relative
// aipersonaggi--

voidparla(Personaggi);
voidaz_parla(Personaggi);
intcontrolla(Oggetti ,Personaggi);
voidinitPers( Insieme<Personaggi>* );
voidpersonaggio_entra();
voidpersonaggio_esce();

// fine modifiche ZAGARIA

```

A ciascuno di queste funzioni è stato aggiunto come parametro Personaggi* e Personaggi* pg, che permette appunto di gestire insieme alle altre funzioni (già esistenti) anche la classe personaggi

```

boolesegui_azione(int, int , Mappa&,Pila<stato_comando>*, bool si,Personaggi* );
// modifica ZAGARIA --aggiuntoilparametroPersonaggi--

voidesegui(int a, Mappa&M, Pila<stato_comando>*,bool si, Personaggi*); // 
// modifica ZAGARIA --aggiuntoilparametroPersonaggi--

voidswitch_enigmi(int a, Mappa&M, Pila<stato_comando>* p, Personaggi* pg); // 
// modifiche ZAGARIA -- aggiuntoilparametropg per lagestionedeipersonaggi

```

- Gioco.cpp

A ciascuno di queste funzioni è stato aggiunto come parametro Personaggi* e Personaggi* pg, che permette appunto di gestire insieme alle altre funzioni (già esistenti) anche la classe personaggi

```
void Gioco::switch_enigmi(int a, Mappa&M, Pila<stato_comando>* p, Personaggi* pg )
// modifica ZAGARIA -- aggiunto il parametro pg per la gestione dei personaggi

void Gioco::esegui(int a, Mappa&M, Pila<stato_comando>* p, bool si, Personaggi* pg )
// modifica ZAGARIA -- modifica della funzione esegui con l'aggiunta del parametro Personaggi

switch_enigmi(a,M,p,pg); // modifica ZAGARIA -- aggiunto il parametro pg
```

Aggiunto il case 43 che inserisce nella storia l'azione da parte del protagonista che ha parlato con il personaggio

```
// iniziamo modifiche ZAGARIA -- aggiunto il case 43 che definisce la comunicazione con un personaggio nel gioco

case 43:

stringa_risposta = "Ha parlato con il personaggio presente nella stanza.";
//aggiungo alla storia

storia_gioco.insStoria(stringa_comando , stringa_risposta);

az_parla(*pg);

break;

// fine modifiche ZAGARIA
```

E' stato aggiunto come parametro Personaggi* e Personaggi* pg, che permette appunto di gestire insieme alle altre funzioni (già esistenti) anche la classe personaggi

```
boolGioco::esegui_azione(int azione, int c2, Mappa&M, Pila<stato_comando>* p, bool si, Personaggi* pg) // modifiche ZAGARIA -- aggiunta alla funzione esegui_azione il parametro Personaggi
```

Aggiunto il parametro pg

```
esegui(cod_azione, M, p, si, pg); // esegue -- modifica ZAGARIA -- aggiunto il parametro pg
```

Inizializzazione dell' insieme personaggi

```
initPers(&insPersonaggi); // modifica ZAGARIA -- inizializzazione dell'insiemePersonaggi
```

Aggiunte funzioni per la gestione dei personaggi

```
// inizio modifiche ZAGARIA -- aggiunge funzioni per la gestione dei personaggi

voidGioco::initPers(Insieme<Personaggi> *insPersonaggi)
{
    Personaggi
    p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23/*, p24, p25, p26, p27, p28, p29, p30*/,
    p31, p32, p33, /*p34, p35, p36, p37, p38, p39, p40, */p41, p42, p43, /*p44, p45, p46, p47,
    p48, p49, p50, */p51, p52, p53, /*p54, p55, p56, p57, p58, p59, p60, */
    p61, p62, p63, /*p64, p65, p66, p67, p68, p69, p70, */p71, p72, p73, /*p74, p75, p76, p77,
    p78, p79, p80, */p81, p82, p83, /*p84, p85, p86, p87, p88, p89, p90, */
    p91, p92, p93/*, p94, p95, p96, p97, p98, p99, p100*/;

    // Luogo 1 1-10
    p1.setNome("Kain");
    p1.setFrasi("Prendila 'Tuta' potrebbe servirti", "La 'Tuta' e' inutile,
    lascialali");
    p1.setLuogo(1);
    insPersonaggi->inserisci(p1);
    p2.setNome("Tom");
}
```

```

    p2.setFrasi("Prendila'Tuta'potrebbeservirti","La'Tuta' e'inutile,
lascialali");
    p2.setOgg(true,51);
    p2.setFrasiconOgg1("Vistochehailatutapuoiandare, qui non
c'e'altrodiimportante","Controlatuttigliindicatori e i pulsanti,
potrebberomostraritiqualcosadiutile");
    p2.setLuogo(1);
    insPersonaggi->inserisci(p2);
    p3.setNome("Jim");
    p3.setFrasi("Prendila'Tuta'potrebbeservirti","La'Tuta' e'inutile,
lascialali");
    p3.setLuogo(1);
    insPersonaggi->inserisci(p3);
    p4.setNome("Adam");
    p4.setFrasi("Prendila'Tuta'potrebbeservirti","La'Tuta' e'inutile,
lascialali");
    p4.setOgg(true,51);
    p4.setFrasiconOgg1("Vistochehailatutapuoiandare, qui non
c'e'altrodiimportante","Controllatuttigliindicatori e i pulsanti,
potrebberomostraritiqualcosadiutile");
    p4.setLuogo(1);
    insPersonaggi->inserisci(p4);
    p5.setNome("Paul");
    .
    .

```

Aggiunto corpo delle funzioni parla, controlla, ecc

```

voidGioco::parla(Personaggi p)
{
    p.stampa();
}

intGioco::controlla(Oggetti oggetti,Personaggi b)
{
    int i;

```

```

i=1;

int lung=oggetti.get_n_oggetti();

bool trovato = false;

if(b.get_ogg2() !=0)

{

    while (i <= lung && !trovato) {

        if (oggetti.get_oggetto(i).get_codice() == b.get_ogg1())

            trovato = true;

        else

            i++;

    }

    if(oggetti.get_oggetto(i).get_luogo()==0)

    {

        i=1;

        trovato=false;

        while (i <= lung && !trovato) {

            if (oggetti.get_oggetto(i).get_codice()

== b.get_ogg2())

                trovato = true;

            else

                i++;

        }

        if(oggetti.get_oggetto(i).get_luogo()==0)

        {

            return 0;

        }

        else

        {

            while (i <= lung && !trovato) {

                if (oggetti.get_oggetto(i).get_codice()

== b.get_ogg1())

                    trovato = true;

                else

                    i++;

            }

            if(oggetti.get_oggetto(i).get_luogo()==0)

```

```

        {
            return 1;
        }
    }
}

else
{
    while (i <= lung && !trovato) {
        if (oggetti.get_oggetto(i).get_codice() ==
b.get_ogg1())
            trovato = true;
        else
            i++;
    }

    if (oggetti.get_oggetto(i).get_luogo()==0)
    {
        return 1;
    }
}
}

void Gioco::personaggio_entra()
{
    if (luogo_attuale!=luogo_precedente&&insPersonaggi.esiste(luogo_attuale)==true)
    {
        pg=insPersonaggi.getElement(luogo_attuale);
        cout<<"Vedo "<<pg.getNome()<<endl;
        presente2=true;
        luogo_precedente=luogo_attuale;
    }
}

void Gioco::personaggio_esce()
{
    if (presente2==true)
    {

```

```

cout<<endl<<pg.getNome ()<<"lasciala stanza"<<endl;
pg.setNome ("");
presente2=false;
}

}

void Gioco::az_parla (Personaggi pg)
{
    int ris_controllo;
    parola2[0]=toupper(parola2[0]);

    if (parola2==pg.getNome ())
    {
        if (pg.getIfNec ()==true)
        {
            ris_controllo=controlla(oggetti,pg);
            if (ris_controllo==0)
                pg.SelezionaFrase2 ();
            elseif (ris_controllo==1)
                pg.SelezionaFrase1 ();
            else
                parla (pg);
        } else
        {
            parla (pg);
        }
    }
    else
        cout <<parola2<<" non e' in questa stanza"<<endl;
}
// fine modifiche ZAGARIA

```

Nella funzione interpreta, eliminata parola2.

```

void Gioco::interpreta(string comando, Mappa&M, Pila<stato_comando>* p, Stato s,
bool si)

{
    int lungh_comando; // lunghezzacomando
    int in; // posizioneinizialediscansionedelcomandoallaricercadiunaparola
    string parola1; // primaparolacomando
    // modifica ZAGARIA -- eliminata --> string parola2; // secondaparolacomando
    int cod_parola2; // codicesecondaparolacomando
    int n1, n2; // variabilitemporanee per costruireilcodicecomando
    bool eseguito; // indicaseazioneeseguita
}

```

Modificati i parametri delle 5 funzioni esegui_azione con l' aggiunta del parametro pg relativo ai personaggi

```

eseguito = esegui_azione(n1 + n2 + cod_parola2, cod_parola2, M,p,si, &pg); // verbo + nome in lu // gosub 500 // modifica ZAGARIA --
modificaparametridellafunzione esegui_azione con l'aggiuntadelparametropg

```

- **Lista.h**

Aggiunta la funzione stampaLista

```

void stampaLista (); //Modifica ZAGARIA -- aggiuntadellafunzione stampaLista --

```

... e il rispettivo corpo di funzione

```

template<class L> void Lista<L>::stampaLista ()
{
    posizione p;
    p = primolista();
}

```

```

while (!finelista(p))
{
    cout<<legglista(p);
    p = succlist(a(p));
}
}

```

5.19 CHIESA

5.19.1 Implementazione modifiche Stato Salute

Come discusso nel Paragrafo 3.19.6.5.1 della documentazione, è stata modificata l'implementazione della classe "StatoSalute" rispecchiante le specifiche descritte. Le modifiche saranno racchiuse

STATOSALUTE.H

Nella parte public sono stati aggiunti i due seguenti metodi:

```

voidSetStatoSalute(int); //AggiornaloStatodi Salute
voidMiracolo(); //Curailpersonaggiodaunaferita

```

STATOSALUTE.CPP

```

voidStatoSalute::SetStatoSalute(int saluteattuale)
{
    Salute=saluteattuale;
}

voidStatoSalute::Miracolo()
{
    if(Ferito()==true) //Precondizione: seseiFerito,
    CancellalaprimaFerita e ripristinalostatodi Salute iniziale
    {
        Listap<Ferita>::posizione pos;
        pos=Ferite.primolista(); //Posizionatisulprimoelemento

        Ferite.canclista(pos); //Cancella l'elemento corrente
        pos=Ferite.succlist(a(pos));
        //Posizionatisulsecondelemento
    }
    if (Ferito()==true) { //seseiancoraferito

```

```

        Salute=90 + rand()%10; //Imposta Salute a un valore
        compreso tra 90 e 99
        PenalitaMosse=1;
    }
    else {
        Salute=100; //Ripristina Salute
        PenalitaMosse=0; //Azzera PenalitàMosse
    }
}

```

Per implementare le modifiche alla classe StatoSalute si è utilizzata la classe Listap che era presente nel progetto senza ulteriori modifiche.

5.19.2 Implementazione luogo "Chiesa" e nuove azioni di gioco

Come discusso nel Paragrafo 3.1 della presente Documentazione, è stato aggiunto un nuovo Luogo all'interno della mappa: la "Chiesa". Per farlo, sono state apportate le seguenti modifiche al file mappa.nav:

- Il numero di Mappe presenti nel Gioco è stato modificato da 19 a 20 per poter contenere il nuovo Luogo;
- Poiché la "Chiesa" è raggiungibile salendo dal "Corridoio Nord", è stato modificato il codice di movimento del "Corridoio":

2,All'estremità nord del corridoio,200316050112,via 1,2,2, via 2,1,1, via 1,6,6;

- È stato aggiunto il nuovo Luogo "Chiesa": 20,In Chiesa,000200000000, via 1,1,1
- Nella cartella Descrizione, è stato aggiunto un nuovo file testuale identificato con 20 contenente la descrizione della "Chiesa".

Inoltre, per poter implementare le nuove funzionalità offerte dal luogo "Chiesa", sono state aggiunte delle nuove azioni all'interno di Astro.h.

```

voidazione_70(); //Azione per confessartinelconfessionale
voidazione_71(); //Azione per sedertisullapanchina
voidazione_72(); //Azionedisupporto all'azione 71
voidazione_73(); //Azione per parlarealsacerdote
voidazione_74(); //Azione per pregare

```

La loro implementazione, inserita all'interno di Astro.cpp, è la seguente:

```

//INIZIO MODIFICA GIOVANNI CASTELLANA

voidAstro::azione_70(){ // azioneconfessati, entraconfessionale
    //Aggiuntacodacasuale
    int n_persone_coda; //Variabiledigeneratorecoda
    string input_utente; //Strinadiappoggio. Contiene l'input
    dell'utente
}

```

```

bool conferma=false; //Confermautente
bool controllo=false; //Variabile di controllo
n_persone_coda=rand()%11; //Genera una variabile casuale compresa tra 0 e

if(n_persone_coda>0) //Se ci sono persone in coda,
chiedi al Giocatore se desidera fare
{
    interfaccia.scrivi_parziale(n_persone_coda);
    interfaccia.scrivi_parziale("persone attendono per
potersi confessare. Desideri aspettare? [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 digitasi,
aggiorna conferma
    }

    conferma=true; //L'utente ha
confermato di volersi confessare
    controllo=true; //Aggiorna il controllo
tempo=tempo-n_persone_coda; //Aggiorna le mosse del giocatore

    interfaccia.a_capo();
    interfaccia.scrivi("----- Attendialcuni istanti...! -----");
    Sleep(850); //Attendialcuni istanti
    interfaccia.a_capo();
    interfaccia.scrivi("----- OTTIMO, TI SEI CONFESSATO! -----");
    interfaccia.scrivi("Hai impiegato");
    interfaccia.scrivi_parziale(n_persone_coda);
    interfaccia.scrivi("unita' di tempo aggiuntive!");
    storia_gioco.insStoria(stringa_comando, "ti sei confessato");
//Aggiorna storia gioco
}
elseif(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
volersi confessare
    controllo=true; //Aggiorna il controllo
    storia_gioco.insStoria(stringa_comando, "non ti sei confessato");
//Aggiorna storia gioco
}
else
{
    interfaccia.scrivi("Non capisco.");
    aggiorna_tempo();
    interfaccia.a_capo();
}

```

```

        interfaccia.scrivi_parziale(n_persone_coda);
        interfaccia.scrivi_parziale("personeattendono per
potersiconfessare. Desideriaspettare? [si/no] ");
        interfaccia.a_capo();
    }
}
else {
    conferma=true; //Se non cisonopersone,
confessatisenzabisognodiconferma
    ("----- OTTIMO, TI SEI CONFESSATO! -----+");
    storia_gioco.insStoria(stringa_comando , "tiseiconfessato");
//Aggiornastoriagioco
}
}

bool controllo=false; //variabile di controllo per fare in modo che la salute
aumenti solo la prima volta che ci si siede

```

```

voidAstro::azione_71(){ // azionesiediti,sieditipanchina

if (controllo==false) { // se non tiseimaiseduto

    if (Salute.GetStatoSalute()<100) { // se il tuo stato di salute
non è ottimale
        int salute_attuale;
        int salute_aggiornata;
        salute_attuale=Salute.GetStatoSalute(); //salvo lo stato di
salute corrente in un'variabile
        salute_aggiornata=salute_attuale+10; //
aggiorno lo stato di salute
        Salute.SetStatoSalute(salute_aggiornata); //
salvo il nuovo stato di salute

        if(salute_aggiornata>100) { // se la salute
aggiornata supera 100
            salute_aggiornata=100; // imposto la salute
aggiornata a 100
            Salute.SetStatoSalute(salute_aggiornata); //
salvo il nuovo stato di salute
        }

        interfaccia.scrivi("----- TI SEI SEDUTO E HAI RECUPERATO
UN PO' DI SALUTE -----");
        interfaccia.scrivi("Salute precedente: ");
    }
}

```

```

                interfaccia.scrivi_parziale(salute_attuale); //stampala
salute precedente
                interfaccia.a_capo();
                interfaccia.scrivi("Salute attuale: ");
                interfaccia.scrivi_parziale(salute_aggiornata); //
stampala salute attuale
                controllo=true;

            }
            else { // seiltuostatodi salute è giàottimale
                interfaccia.scrivi("----- TI SEI SEDUTO E IL TUO STATO
DI SALUTE E' OTTIMO -----");
                controllo=true;
            }
        }
        else { // segìatiseiseduto, non tipermettedirecuperare salute
            interfaccia.scrivi("----- TI SEI SEDUTO MA NON HAI RECUPERATO
SALUTE -----");
        }

        storia_gioco.insStoria(stringa_comando , "tiseiseduto");
//Aggiornastoriagioco

    }

voidAstro::azione_72(){ // azione che serve a impostare la variabile
controllo utile nell'azione 71, quando si esce dalla chiesa e si va a sud
(unica direzione consentita)

    luogo_attuale=2;
    controllo=false;
    storia_gioco.insStoria(stringa_comando , "seiandato
all'estremità 'norddelcorridoio'); //Aggiornastoriagioco
}

voidAstro::azione_73(){ //azioneparlasacerdote
    interfaccia.scrivi("----- Haiparlatocoil sacerdote e ti ha
suggeritodipregare, potrebbe accadere un miracolo! -----");
    storia_gioco.insStoria(stringa_comando ,
"haiparlatocoil sacerdote'); //Aggiornastoriagioco
}

voidAstro::azione_74(){ //azione PREGA

    Listap<Ferita> ferite; //FeritecontrattatedalGiocatore
}

```

```

if(Salute.Ferito()==false) //Se non sei ferito
    interfaccia.scrivi("Il tuo stato di Salute è ottimo!");
else
{
    ferite=Salute.Get_Ferite(); //
aggiorna le ferite contrattate dal giocatore
    Listap<Ferita>::posizione pos;
    pos=ferite.primoLista(); //Posizionati sul primo elemento

    int casuale=rand()%10; //genera un numero casuale da 0 a 9

    if (casuale<5) { // se il numero casuale è minore di 5
        MIRACOLO: -----+ HAI PREGATO ED E' AVVENUTO UN
        interfaccia.scrivi("-----+");
        interfaccia.a_capo();
        interfaccia.scrivi("Sei stato curato da");
        interfaccia.scrivi_parziale(ferite.leggiLista(pos).GetNome());
        //Stampa il nome della ferita
        Salute.Miracolo(); //invoca la procedura Miracolo di
        StatoSalute
        pos=ferite.succLista(pos); // Posizionati sull'elemento
        successivo
    }
    else { // se il numero casuale è maggiore di 5
        interfaccia.scrivi("----- OTTIMO, HAI PREGATO! -----+");
    }
}

storia_gioco.insStoria(stringa_comando , "ha ipregato");
//Aggiorna storia gioco
}

//FINE MODIFICHE GIOVANNI CASTELLANA

```

Infine, per rendere effettive le modifiche, è stata apportata la seguente aggiunta in Astro.cpp

```

...
case 70:
    azione_70();
    break;

case 71:
    azione_71();
    break;

```

```

case 72:
    azione_72();
    break;

case 73:
    azione_73();
    break;

case 74:
    azione_74();
    break;
...

```

5.20 UFFICIO / DEPOSITO

5.20.1 Implementazione consegna documento

```

//AZIONE CONSEGNA DOCUMENTO
void Astro::azione_140 ()
{
    int doc;
    int doc1;
    int i;
    bool trovato;
    if (luogo_attuale==12)
    {
        interfaccia.scrivi("Hai questo documento: ");
        primo_doc.elenca_documenti(0);
        interfaccia.scrivi ("Digita il codice del documento se vuoi
lasciarlo\n");
        cin >> doc;
        for (i=0 ; i < primo_doc.get_n_documenti() ; i++)
        {
            if (doc == primo_doc.get_codice_documento(i))
            {
                if(primo_doc.get_luogo_documento(i)== 0)
                {
                    primo_doc.set_luogo_documento(i, 12);
                    interfaccia.scrivi ("Documento lasciato!\n");
                    trovato = true;
                    ritirato = false;
                    break;
                }
            }
            else
                trovato=false;
        }
        if (trovato == false)
        {
            interfaccia.scrivi ("Non eri in possesso del documento");
        }
        if (doc > 20)
        {
            interfaccia.scrivi ("Il codice del documento che hai digitato non
esiste");
        }
    }
}

```

```
    }  
}
```

5.20.2 Implementazione ritira documento

```
//AZIONE RITIRA DOCUMENTO  
void Astro::azione_141 ()  
{  
    int k;  
    int doc;  
  
    primo_doc.stampa_nomi_documenti(12);  
  
    interfaccia.scrivi("Digita il codice del documento che vuoi ritirare ");  
    cin >> doc;  
  
    if (ritirato == false)  
    {  
        for (k=0; k<primo_doc.get_n_documenti(); k++)  
        {  
            if (doc == primo_doc.get_codice_documento(k))  
            {  
                if (primo_doc.get_luogo_documento(k) == 12)  
                {  
                    primo_doc.set_luogo_documento(k, 0);  
                    interfaccia.scrivi("Hai preso il documento scelto.");  
                    ritirato = true;  
                    break;  
                }  
            else  
                interfaccia.scrivi("Questo documento ce l'hai gia'!");  
            break;  
        }  
    }  
    else  
        interfaccia.scrivi("Hai gia' un documento nell'inventario!");  
}
```

5.21 SALI&TABACCHI

Per poter aggiungere il nuovo luogo sono stati modificati i seguenti file:

Mappa.nav
43.txt
Astro.h
Astro.cpp
Gioco.h
Gioco.cpp
Interfaccia.h
Interfaccia.cpp

È stata aggiunta la classe:

Scontrino.h

Scontrino.cpp

5.21.1 Mappa.nav

Le modifiche effettuate al file Mappa.nav riguardano l'aggiunta del luogo 43 "Sali&Tabacchi", la cui strada per arrivare è:

43,Nel Sali&Tabacchi,0000250000,via 1,2,2

È stato cambiato il percorso dell'Aula:

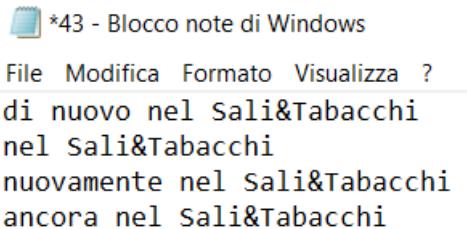
Da 25,Nell'Aula,042600000000,via 1,1,1 a 25,Nell'Aula,042600430000,via 1,1,1

Sono state aggiunte le seguenti righe per il controllo del movimento:

25,43,via 1,ovest,4,4,2,2
43,25,via 1,est,4,4,2,2

5.21.2 43.txt

Il file di testo "43.txt" nella cartella descrizioni è stato aggiornato inserendo la seguente descrizione:



*43 - Blocco note di Windows
File Modifica Formato Visualizza ?
di nuovo nel Sali&Tabacchi
nel Sali&Tabacchi
nuovamente nel Sali&Tabacchi
ancora nel Sali&Tabacchi

5.21.3 Astro.h

In Astro.h sono state aggiunti:

- I metodi, relativi alle azioni che è possibile effettuare nel Sali&Tabacchi:
- `//inizio modifiche Lentini Giovanni`
- `void azione_146(); //leggi-guarda-vedi orari`
- `void azione_142(); //gioca lotteria`
- `void azione_143(); //compra-acquista-prendi sale`
- `void azione_144(); //compra-acquista-prendi sigarette`
- `void azione_145(); //buonus-usa elaboratore(bonus)`
- `//fine modifiche Lentini Giovanni`

- La dichiarazione delle variabili che verranno utilizzate:

```

- //inizio modifiche Giovanni Lentini
-     scontrino ricevuta;
-     listaVettore<scontrino> acquisti;
-     typename listaVettore<scontrino>::posizione punt=NULL;
-     Pila<int> lotteria;
-     int num_biglietti=0;
- //fine modifiche Giovanni Lentini

```

5.21.4 Astro.cpp

Le modifiche effettuate al file Astro.cpp sono:

- Aggiunta dei vocaboli nel dizionario:

```

- //inizio modifiche Giovanni Lentini
-         vocabolario.inserisci("Sali&Tabacchi",16);
-         vocabolario.inserisci("gioca",100);
-         vocabolario.inserisci("acquista",33);
-         vocabolario.inserisci("sigarette",101);
-         vocabolario.inserisci("pacchetto sigarette",101);
-         vocabolario.inserisci("sale",102);
-         vocabolario.inserisci("orari",103);
-         vocabolario.inserisci("lotteria",105);
-         vocabolario.inserisci("vedi", 106);
-         vocabolario.inserisci("ottieni",107);
-         vocabolario.inserisci("bonus",104);
-         vocabolario.inserisci("elaboratore",104);
-         vocabolario.inserisci("elaboratore bonus",104);
- //fine modifiche Giovanni Lentini

```

- Aggiunta degli oggetti presenti nel Sali&Tabacchi:

```
//inizio modifiche Lentini Giovanni
-     oggetti.inserisci(Oggetto("gli orari", 103,-43));
-     oggetti.inserisci(Oggetto("una lotteria", 105,-43));
-     oggetti.inserisci(Oggetto("del sale", 102,43));
-     oggetti.inserisci(Oggetto("delle sigarette", 101,43));
-     oggetti.inserisci(Oggetto("un elaboratore bonus", 104,-43));
- //fine modifiche Lentini Giovanni

```

- Inserimento delle azioni previste per il luogo Sali&Tabacchi:

```
- //inizio modifiche Giovanni Lentini
-     azioni.inserisci(4300100103,146); //guarda orari
-     azioni.inserisci(4300250103,146); //leggi orari
-     azioni.inserisci(4301060103,146); //vedi orari
-     azioni.inserisci(4301000105,142); //gioca lotteria
-     azioni.inserisci(4300080102,143); //prendi sale
-     azioni.inserisci(4300330102,143); //acquista sale
-     azioni.inserisci(4300330102,143); //compra sale
-     azioni.inserisci(4300080101,144); //prendi sigarette
-     azioni.inserisci(4300330101,144); //acquista sigarette
-     azioni.inserisci(4300330101,144); //compra sigarette
-     azioni.inserisci(4301070104,145); //ottieni bonus
-     azioni.inserisci(4300290104,145); //usa elaboratore
-     azioni.inserisci(4300290104,145); //usa elaboratore bonus
-     azioni.inserisci(4300800104,145); //usa elaboratore
-     azioni.inserisci(4300800104,145); //usa elaboratore bonus
-     //fine modifiche Lentini Giovanni
```

- Aggiunta delle azioni:

```
- //inizio modifiche Lentini Giovanni
-
-     case 142: //gioca lotteria
-     azione_142();
-     break;
-     case 143:
-     azione_143(); //compra-acquista-prendi sale
-     break;
-     case 144: //compra-acquista-prendi sigarette
-     azione_144();
-     break;
-     case 145: //usa elabortore (bonus) - ottieni bonus
-     azione_145();
-     break;
-     case 146:
-         azione_146(); //leggi orari
-         break;
-     //fine modifiche Lentini Giovanni
```

5.21.4.1Azione_146

L'azione_146 consente al giocatore di poter visualizzare gli orari.

5.21.4.2 Azione_142

L'azione_142 permette al giocatore di giocare alla lotteria. Per la gestione dei biglietti della lotteria è stata usata la struttura dati della pila con vettore.

```
void Astro::azione_142(){//gioca lotteria
bool flag;
bool controllo_numeri = false;
int numero_scelto;
string risposta;
int numero_estrazione;

bool verifica_portafoglio = portafoglio.hai_Portafoglio(oggetti);

if(verifica_portafoglio){
system("cls");

interfaccia.scrivi("\n\t\t%%%%%%%%%%%%%\n");
interfaccia.scrivi("\t\t%%%%%%%%%%%%%\n");
LOTTERIA NEUTRONIA
interfaccia.scrivi("\t\t%%%%%%%%%%%%%\n");
interfaccia.scrivi("\t\t%%%%%%%%%%%%%\n");
interfaccia.scrivi("\t\t%%%%%%%%%%%%%\n");
Istruzioni per giocare:
%%%%%%%%%%%%%\n");
```


5.21.4.3Azione 143

L'azione_143 permette al giocatore di poter acquistare il prodotto "sale". Nel progetto da integrare tale acquisto era stato gestito con la struttura dati delle coda con vettore. Ho ritenuto opportuno, considerando che non vi è nessun vincolo, nella funzione, sull'inserimento degli acquisti, di cambiare la struttura dati in quella della lista con vettore.

```

void Astro::azione_143(){ //compra-acquista-prendi sale

    bool verifica_denaro = false;
    bool flag = false;
    bool verifica_risp = false;
    string risposta;
    string risp;
    int num_persone = rand()%11+1;
    int conta = 0; // conta quanti prodotti sono stati acquisati nella stessa fila

    interfaccia.scrivi("\nCi sono");
    interfaccia.scrivi_parziale(num_persone);
    interfaccia.scrivi(" persone in coda per acquistare il sale.");
    interfaccia.scrivi("Il costo di un pacco di sale e' 2 euro.");
    interfaccia.scrivi("Desideri aspettare e pagare? [s/n]:");

    do{
        cin>>risposta;
        if ((risposta == "s") || (risposta == "si")){
            tempo = tempo - num_persone;

            //azioni
            verifica_denaro = verifica_den(2.0f);

            if(verifica_denaro){
                do{
                    conta++;

                    interfaccia.scrivi("\n *****\n");
                    interfaccia.scrivi(" * Prodotto acquistato! *");
                    interfaccia.scrivi("\n *****\n");

                    ricevuta.set_articolo("sale");
                    ricevuta.set_prezzo(2);
                    acquisti.insLista(ricevuta, punt);

                    cout<<endl;
                    interfaccia.scrivi("Vuoi acquistare altro sale? [s/n]");
                do{
                    cin>>risp;

                    if ((risp!="s") && (risp!="si") && (risp!="n") && (risp!="no")){
                        cout<<"\nRisposta non accettata, inseriscila nuovamente"<<endl;
                        verifica_risp= true;
                    }
                }else{
                    verifica_risp= false;
                }

                } while(verifica_risp);

                if((risp=="s") || (risp=="si")){
                    verifica_denaro = verifica_den(2);}
            }while(((risp=="s") || (risp=="si")) && ( verifica_denaro == true)&& (verifica_risp== false));
        }
    }
}

```

```

cout << "\n\n- Premi <invio> per stampare lo scontrino -\n";
interfaccia.attendi();

    interfaccia.scrivi("Sto stampando lo scontrino ");
    Sleep(300);
    cout<< " .";
    Sleep(300);
    cout<< " .";
    Sleep(300);
    cout<< " . " << endl;
system("cls");

interfaccia.scrivi("\n.....\n");
    interfaccia.scrivi("\n..      +++      +++++++\n");
..\\n");
    interfaccia.scrivi("\n..      ++      ++\n");
..\\n");
    interfaccia.scrivi("\n..      +      +++++\n");
..\\n");
    interfaccia.scrivi("\n..      ++      +++\n");
..\\n");
    interfaccia.scrivi("\n..      +++      +++++\n");
..\\n");

interfaccia.scrivi("\n.....\n");
    interfaccia.scrivi("\n..      articolo: sale\n");
..\\n");
    interfaccia.scrivi("\n..      costo: 2 euro\n");
..\\n");

interfaccia.scrivi("\n.....\n");

    if(conta >1){
        cout<<"\n Sono stati stampati "<<conta<<" scontrini del prodotto
'sale'"<<endl;
        cout<<"\n Hai speso "<<conta*2<< " euro\n\n"<< endl;
    }
}
flag = true;
}

if ((risposta == "n") || (risposta == "no")){
    flag = true;
    cout<<"\nArrivederci =) !"\n";
}
if(flag != true){
    cout<<"\nRisposta non accettata, inseriscila
nuovamente"\n";
}
}while( flag != true);
}

```

5.21.4.4Azione_144

L'azione_144 permette al giocatore di poter acquistare il prodotto "sigarette". Come nell'azione precedente, anche questo acquisto è stato gestito con la coda con vettore. Tale acquisto è stato implementato nello stesso modo dell'azione precedente, quindi, per lo stesso motivo spiegato prima, ho

ritenuto opportuno anche in questo caso sostituire la struttura dati utilizzata nel progetto integrativo con la lista con vettore.

```
void Astro::azione_144(){ //compra-acquista-prendi sigarette

    bool verifica_denaro = false;
    bool flag = false;
    bool verifica_risp = false;
    string risposta;
    string risp;
    int num_persone = rand()%11+1;
    int conta = 0;

    interfaccia.scrivi("\nCi sono");
    interfaccia.scrivi_parziale(num_persone);
    interfaccia.scrivi(" persone in coda per acquistare le sigarette.");
    interfaccia.scrivi("Il costo di un pacchetto di sigarette e' 5 euro.");
    interfaccia.scrivi("Desideri aspettare e pagare? [s/n]:");

    do{
        cin>>risposta;
        if ((risposta == "s") || (risposta == "si")){
            tempo = tempo - num_persone;
            //azioni
            verifica_denaro = verifica_den(5);

            if(verifica_denaro){
                do{
                    conta++;

                    interfaccia.scrivi("\n *****\n");
                    interfaccia.scrivi(" * Prodotto acquistato! *");
                    interfaccia.scrivi("\n *****\n");

                    ricevuta.set_articolo("sigarette");
                    ricevuta.set_prezzo(5);
                    acquisti.insLista(ricevuta, punt);
                }
            }
        }
    cout<<endl;

    interfaccia.scrivi("Vuoi acquistare un altro pacchetto di sigarette? [s/n]");
    do{
        cin>>risp;

        if ((risp!="s") && (risp!="si") && (risp!="n") && (risp!="no")){
            cout<<"\nRisposta non accettata, inseriscila nuovamente"<<endl;
            verifica_risp= true;
        }
        else{
            verifica_risp= false;
        }
    }

    if((risp=="s") || (risp=="si")){
        verifica_denaro = verifica_den(5);}
```

```

}while(((risp=="s") || (risp=="si")) && ( verifica_denaro == true));
cout << "\n\n- Premi <invio> per stampare lo scontrino -\n";
interfaccia.attendi();

    interfaccia.scrivi("Sto stampando lo scontrino ");
    Sleep(300);
    cout<<".";
    Sleep(300);
    cout<<".";
    Sleep(300);
    cout<<".\n" << endl;
system("cls");

interfaccia.scrivi("\n.....\n");
    interfaccia.scrivi("\n..      +++      ++++++++\n");
..\\n");
    interfaccia.scrivi("\n..      ++      ++      ++\n");
..\\n");
    interfaccia.scrivi("\n..      +      +++++      ++\n");
..\\n");
    interfaccia.scrivi("\n..      ++      +++      +++      ++\n");
..\\n");
    interfaccia.scrivi("\n..      +++      +++++      ++\n");
..\\n");

interfaccia.scrivi("\n.....\n");
    interfaccia.scrivi("\n..      articolo: sigarette\n");
..\\n");
    interfaccia.scrivi("\n..      costo: 5 euro\n");
..\\n");

interfaccia.scrivi("\n.....\n");

    if(conta >1){
        cout<<"\n\n\t Sono stati stampati "<<conta<<" scontrini del prodotto
'sigarette'"<<endl;

        cout<<"\n\n\t Hai speso " <<conta* 5<< " euro\n\n" << endl;
    }
}
flag = true;
}
if ((risposta == "n") || (risposta == "no") ){
    flag = true;
    cout<<"\nArrivederci =) !"\n" << endl;
}
if(flag != true){
    cout<<"\nRisposta non accettata, inseriscila
nuovamente"\n" << endl;
}
}while( flag != true);

}

```

5.21.4.5Azione_145

L'azione_145 permette al giocatore di poter utilizzare l'elaboratore bonus. Questo permette la conversione degli scontrini in tempo bonus, in più, si ha la possibilità di poter giocare nuovamente l'ultimo biglietto della lotteria acquistato per ottenere tempo bonus.

```
void Astro::azione_145(){ // buonus-usa elaboratore(bonus)

float totale;
float time;
bool flag = false;
bool flag_risposta=false;
int scelta;
int num;
int numeros = 0;
string risposta;

system("cls");

interfaccia.scrivi("\n\n.....\n");
;

interfaccia.scrivi(".....\n");
    interfaccia.scrivi(".....          BENVENUTO NELL'ELABORATORE BONUS
.....\n");
    interfaccia.scrivi(".....\n");
    interfaccia.scrivi(".....      QUI PUOI CONVERTIRE LE RICEVUTE DEI
.....\n");
    interfaccia.scrivi(".....      TUOI ACQUISTI EFFETUATI NEL SALI&TABACCHI
.....\n");
    interfaccia.scrivi(".....          IN TEMPO BONUS !
.....\n");

    interfaccia.scrivi(".....      Il valore di ogni scontrino piu' un unita' a
.....\n");

    interfaccia.scrivi(".....      sorpresa vengono aggiunti come secondi
.....\n");

    interfaccia.scrivi(".....      bonus del tempo redsiduo a disposizione
.....\n");
    interfaccia.scrivi(".....      oppure puoi provare a vincere 20 secondi
.....\n");
    interfaccia.scrivi(".....      bonus con i biglietti comprati alla lotteria
.....\n");
    interfaccia.scrivi(".....\n");
    interfaccia.scrivi(".....\n");

    interfaccia.scrivi(".....      Vuoi ricavare secondi bonus?[s/n]
.....\n");
do{
    cin>>risposta;

    if((risposta=="s") ||(risposta == "si")){
        flag_risposta=false;
```



```

        interfaccia.scrivi(".....      Adesso il tuo tempo residuo e':
.....\n");                      cout<<"\n.....          "<

```

```

        tempo = tempo + totale ;
            Sleep(700);
            interfaccia.scrivi(".....\n");
            cout<<".....Tempo residuo dopo la conversione del "<<i+1<< " scontrino.....\n"<<endl;
            cout<<"\n.....\n";
            interfaccia.scrivi("\n.....\n");
            interfaccia.scrivi(".....\n");
            interfaccia.scrivi(".....\n");

            Sleep(1000);

    numeros++; // conta quanti scontrini sono stati convertiti
    flag = false;

}

else{
    flag = true;
}
} //fine for

if (flag == true){

    interfaccia.scrivi("..... Non hai piu' scontrini da convertire! .....");
    cout<<"..... Sono stati convertiti solo "<<numeros<< " scontrini\n";
    interfaccia.scrivi(".....\n");
    interfaccia.scrivi(".....\n");

}

cout<<"..... Tempo prima della conversione: "<<time<< "\n";
cout<<"..... Tempo dopo la conversione: "<<tempo-1<< "\n";

break;

case 3:

    //tenta ancora la fortuna con i biglietti giocati alla lotteria

interfaccia.scrivi("\n.....\n");

    if(lotteria.pilavuota()){

        interfaccia.scrivi("..... Non hai nessun biglietto della lotteria\n");
        interfaccia.scrivi("..... per tentare la fortuna\n");
    }

    else{
        cout<<"..... Hai acquistato "<<num_biglietti<< " biglietti della lotteria\n";
    }
}

```

```

        cout<<"..... L'ultimo biglietto comprato era il Numero : "
<<lotteria.leggipila()<< ".....\n"<<endl;

        cout << "\n\n- Premi <invio> per scoprire se hai vinto -\n";
interfaccia.attendi();

        if((lotteria.leggipila() % 2 == 0) || (lotteria.leggipila() % 20 == 0) ||
(lotteria.leggipila() % 6 == 0) || (lotteria.leggipila() % 15 == 0)){
            interfaccia.scrivi("\n..... Complimenti hai vinto un bonus di 20 secondi!
.....\n");

interfaccia.scrivi(".....\n");

            time = tempo;
            tempo = tempo + 20 ;
            cout<<"..... Tempo prima della vincita: "<<time<< "
.....\n"<<endl;
            cout<<"..... Tempo dopo la vincita: "<<tempo-1<< "
.....\n"<<endl;

        }

else{    interfaccia.scrivi("\n..... Mi dispiace non hai vinto
.....\n");
        }
lotteria.fuoripila();
    }
    break;

case 4:
    //esci elaboratore
    break;

default:
    cout<<"Mi dispiace ma questa scelta non e' compresa, inseriscila nuovamente";
}
} while((scelta!= 1) && (scelta!= 2) && (scelta!= 3)&& (scelta!= 4));

interfaccia.scrivi(".....\n");

interfaccia.scrivi("..... ARRIVEDERCI E GRAZIE PER AVER ..... \n");
interfaccia.scrivi("..... USATO L'ELABORATORE BONUS! ..... \n");

interfaccia.scrivi(".....\n");
interfaccia.scrivi(".....\n");

} //fine esito riposta = si

else if((risposta=="n") || (risposta=="no")){
    flag_risposta=false;

    interfaccia.scrivi(".....\n");
    interfaccia.scrivi("..... ARRIVEDERCI E GRAZIE PER AVER ..... \n");
    interfaccia.scrivi("..... USATO L'ELABORATORE BONUS! ..... \n");
    interfaccia.scrivi(".....\n");
    interfaccia.scrivi(".....\n");

}

if ((risposta!="s") && (risposta!="si") && (risposta!="n") && (risposta!="no")){
    flag_risposta = true;
}

```

```

interfaccia.scrivi("..... .....\\n");
interfaccia.scrivi(".....Mi dispiace ma la risposta inserita non e' valida!....\\n");
interfaccia.scrivi(".....    Perfavore, inserisci nuovamente la scelta .....\\n");
interfaccia.scrivi("..... .....\\n");
}

}while (flag_risposta== true);
}

```

5.21.5 Gioco.h

In gioco.h è stato inserito il seguente metodo per il controllo del denaro posseduto:

```
bool verifica_den(int);
```

5.21.6 Gioco.cpp

In gioco.cpp è stato svolto il seguente controllo:

```

//inizio modifiche Lentini Giovanni
bool Gioco::verifica_den(int elemento){

    int prezzo_elemento;
    float saldo = 0;

bool verifica = portafoglio.hai_Portafoglio(oggetti);

switch(elemento){
    case 2:
        prezzo_elemento = 2;
        break;

    case 5:
        prezzo_elemento = 5;
        break;

    default:
        prezzo_elemento=0;
    }

if(verifica){
    saldo = portafoglio.get_contanti();

    if(saldo >= prezzo_elemento){
        saldo = saldo - prezzo_elemento;
        portafoglio.set_contanti(saldo);
    }

    else{
        interfaccia.scrivi("\nNon hai contanti a sufficienza per acquistare il
prodotto");
        verifica = false;
    }
}
else{

```

```

        switch(elemento){
            case 2:
                interfaccia.scrivi("\nPer poter effettuare acquisti devi avere il portafoglio
con almeno 2 euro all'interno ");
                verifica = false;
                break;

            case 5:
                interfaccia.scrivi("\nPer poter effettuare acquisti devi avere il portafoglio
con almeno 5 euro all'interno ");
                verifica = false;
                break;

            default:
                interfaccia.scrivi("\nPer poter effettuare acquisti devi avere il portafoglio
con del denaro all'interno ");
                verifica = false;
            }
        }

        return verifica;
    }

//fine modifiche Lentini Giovanni

```

5.21.7 Interfaccia.h

Nel file interfaccia.h è stata inserita la seguente procedura:

```
void attendi();
```

5.21.8 Interfaccia.cpp

Nel file interfaccia.cpp è stata eseguita la seguente procedura:

```

//inizio modifiche Lentini Giovanni
void Interfaccia::attendi(){
string invio;
fflush(stdin);

getline(cin, invio);
system("cls");

cout << endl;
}

//fine modifiche Lentini Giovanni

```

5.21.9 Scontrino.h

Implementazione della classe scontrino:

```
// Fine modifiche Lentini Giovanni
```

```
#ifndef SCONTRINO_H
#define SCONTRINO_H
#include <string>
#include <iostream>
```

```

using namespace std;

class scontrino
{
public:
    scontrino();
    ~scontrino();
    float get_prezzo();
    string get_articolo();
    void set_prezzo(float);
    void set_articolo(string);

private:
    float prezzo;
    string articolo;
};

#endif // SCONTRINO_H

// Fine modifiche Lentini Giovanni

```

5.21.10 Scontrino.cpp

```

//Inizio modifiche Lentini Giovanni

#include "scontrino.h"

scontrino::scontrino() {}

scontrino::~scontrino() {}

float scontrino::get_prezzo(){ return prezzo; }

string scontrino::get_articolo(){ return articolo; }

void scontrino::set_prezzo(float p){ prezzo = p; }

void scontrino::set_articolo(string a){ articolo = a; }

// Fine modifiche Lentini Giovanni

```

5.21.11 Risoluzione Bug Del Portafoglio

Durante l'esecuzione del programma è sorto un problema sulla raccolta dei soldi nelle varie stanze presenti nel gioco. Nello specifico, anche se il giocatore possedeva il portafoglio, non riusciva a raccogliere le varie banconote presenti nell'ambiente di gioco. Questo non permetteva di verificare i controlli implementati per il corretto funzionamento del Sali&Tabacchi. Dopo un'attenta analisi, sono arrivata alla conclusione che il problema derivasse dall'implementazione della slot machine e dei suoi oggetti. Infatti, questi ultimi che dovrebbero rappresentare le puntate (5 euro, 10 euro, ecc.) hanno la stessa etichetta delle banconote presenti nel gioco, ma codice oggetto diverso. Per risolvere tale problema sono state effettuate le seguenti modifiche:

- Rimozione dei vocaboli riguardanti le portate dal dizionario:

```

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);

```

- Modifica dei codici delle puntate in modo da farli coincidere con i codici delle banconote presenti nel gioco:

```

oggetti.inserisci(Oggetto("una slotmachine", 86,-35));

oggetti.inserisci(Oggetto("5 euro",3,-99)); //Luogo -99 in quanto non visibili
oggetti.inserisci(Oggetto("10 euro",4,-99));//modifiche Lentini, modificato i codici
oggetto per risolvere il bug del portafoglio
oggetti.inserisci(Oggetto("20 euro",5,-99));
oggetti.inserisci(Oggetto("50 euro",6,-99));
oggetti.inserisci(Oggetto("100 euro",94,-99));

```

5.22 DISTANZA/SFORZO

Seguono le integrazioni eseguite, in base alle modifiche apportate precedentemente dal collega De Rinaldis, nei file prima citati.

5.22.1 Contapassi.cpp

Righe 10-16 (Definito svuotamento delle pile. Tratto da De Rinaldis)

```

10 // (inizio integrazione Spina)
11 while(!traccia_passi.pilavuota())
12     traccia_passi.fuoripila();
13
14 while(!calcola.pilavuota())
15     calcola.fuoripila();
16 // (fine integrazione Spina)

```

Righe 36-39 (Spostato da oggetti privati in Contapassi.h a oggetti locali della funzione StampaPassi. Tratto da De Rinaldis)

```

36 // (inizio integrazione Spina)
37 Pila<string> aus_traccia_passi;
38 Pila<int> aus_calcola;
39 // (fine integrazione Spina)

```

Riga 43 (Chiarifica del messaggio. Tratta da De Rinaldis)

```

43 cout<<"** Ti sei spostato "<<numer_cont<<" volta/e dall'ultima richiesta di inizio del conteggio."<<endl; // (integrazione Spina)

```

Riga 57 (Aggiunta del secondo parametro di condizione, per risolvere il bug del loop infinito delle stampe, prima presente. Tratto da De Rinaldis)

```

57 }while(stampa<=0 && numer_cont != 0); // (integrazione Spina)

```

5.22.2 Mappa.h

Righe 52-54 (Aggiunta metodo per l'acquisizione della lunghezza tra due aree. Tratto da De Rinaldis)

```
52 // (inizio integrazione Spina)
53 float get_lunghezza(int, int); //data il nodo precedente e successivo restituisce il peso dell'area
54 // (fine integrazione Spina)
```

5.22.3 Mappa.cpp

Righe 133-135 (Modifica chiamata del metodo per l'acquisizione del nodo. Tratto da De Rinaldis)

```
133 // (inizio integrazione Spina)
134 if (temp.getNode() != fittizio->getNode())
135 // (fine integrazione Spina)
```

Righe 154-156 (Modifica chiamata del metodo per l'acquisizione del nodo. Tratto da De Rinaldis)

```
154 // (inizio integrazione Spina)
155 if (temp.getNode() != fittizio->getNode())
156 // (fine integrazione Spina)
```

Righe 370-386 (Modifica chiamate dei metodi per l'acquisizione del nodo. Tratto da De Rinaldis)

```
370 // (inizio integrazione Spina)
371 if (! (arrivo.getNode() == fittizio->getNode()))
372 {
373
374     char orizz=205; //carattere per la stampa delle cornici
375     system("cls");
376     interfaccia.scrivi("Inzagitto :");
377     for (int i=0; i<100; i++) interfaccia.scrivi_carattere(orizz);
378     if (arrivo.getNode() == fittizio->getNode())
379         interfaccia.scrivi("Non hai definito il luogo di arrivo, impossibile calcolare il percorso ...");
380     interfaccia.a_capo();
381     if (arrivo.getNode() == partenza.getNode())
382         interfaccia.scrivi("Il luogo di partenza coincide con quello di arrivo, non e' necessario continuare !");
383     interfaccia.a_capo();
384     bool problemi=(partenza.getNode() == fittizio->getNode())
385         || arrivo.getNode() == fittizio->getNode() || arrivo.getNode() == partenza.getNode();
386 // (fine integrazione Spina)
```

Righe 537-539 (Modifica chiamata del metodo per l'acquisizione del nodo. Tratto da De Rinaldis)

```
537 // (inizio integrazione Spina)
538 if (i.getNode() != r.getNode()) { // se il nodo corrente non e' r
539 // (fine integrazione Spina)
```

Righe 575-577 (Modifica chiamata del metodo per l'acquisizione del nodo. Tratto da De Rinaldis)

```
575 // (inizio integrazione Spina)
576 if (T[k].legginodo().getNode() == j.getNode()) {
577 // (fine integrazione Spina)
```

Righe 639-652 (Modifica chiamate del metodo per l'acquisizione del nodo. Tratto da De Rinaldis)

```

639           // (inizio integrazione Spina)
640           if (T[k].legginodo () ==d.getNodo ()) trovato=true;
641           else k++;
642       }
643       while (k>0) //mentre non sono arrivato alla radice o ho trovato un nodo senza predecessore
644       {
645           P.inpila(T[k].legginodo()); //innila il nodo
646           k=T[k].leggipredecessore(); //salvo al predecessore
647       }
648       if (k==0) //se mi sono fermato alla radice (l'ho saltata)
649       {
650           P.inpila(T[k].legginodo()); //innilo la radice
651       }
652   // (fine integrazione Spina)

```

Righe 1743-1745 (Modifica chiamata del metodo per l'acquisizione del nodo. Tratto da De Rinaldis)

```

1743           // (inizio integrazione Spina)
1744           if (templ.getNodo () != fittizio->getNodo ()) {
1745               // (fine integrazione Spina)

```

Righe 1827-1829 (Modifica chiamata del metodo per l'acquisizione del nodo. Tratto da De Rinaldis)

```
1827 // (inizio integrazione Spina)
1828 if (templ.getNode() != fittizio->getNode()) {
1829 // (fine integrazione Spina)
```

Righe 2111-2116 (Dichiarazione metodo per l'acquisizione della lunghezza tra due aree. Tratta da De Rinaldis)

```
2111 // (inizio integrazione Spina)
2112 float Mappa::get_lunghezza(int p, int s){ //restituisce la lunghezza dell'arco passando l'id del nodo di partenza e di arrivo
2113
2114 return mappa->leggiarco(nodo_da_id(p),nodo_da_id(s)).leggilunghezza();
2115 }
2116 // (fine integrazione Spina)
```

5.22.4 Gioco.h

Riga 58 (Inclusione classe del calcolo dei passi. Tratto da De Rinaldis)

```
58 #include "Contapassi.h" // (integrazione Spina)
```

Righe 231-236 (Aggiunta variabili per il calcolo dei passi. Tratto da De Rinaldis)

```
231 // (inizio integrazione Spina)
232 contapassi sforzo;
233 int distanza;
234 string luogo_succ;
235 string luogo_prec;
236 // (fine integrazione Spina)
```

5.22.5 Gioco.cpp

Righe 1099-1104 (Aggiunta della distanza tra le due aree al contapassi. Tratto da De Rinaldis)

```
1099 // (inizio integrazione Spina)
1100 int distanza = M.get_lunghezza(luogo_attuale, a);
1101 sforzo.ins_distanza(distanza);
1102 luogo_prec = M.get_nome_luogo(a);
1103 luogo_succ = M.get_nome_luogo(luogo_attuale);
1104 // (fine integrazione Spina)
```

Righe 1108-1110 (Aggiunta dell'area di partenza e di arrivo al contapassi. Tratto da De Rinaldis)

```
1108 // (inizio integrazione Spina)
1109 sforzo.ins_contapassi(luogo_prec, luogo_succ);
1110 // (fine integrazione Spina)
```

5.22.6 Astro.h

Righe 278-280 (Aggiunta metodo per la visualizzazione delle informazioni sulla distanza/sforzo, successivamente sostituito da "azione_141" ad "azione_147")

```
278 // (inizio modifiche SPINA)
279 void azione_147(); // stampa info su sforzo del personaggio
280 // (fine modifiche SPINA)
```

5.22.7 Astro.cpp

Righe 481-483 (Inserimento del termine *sforzo* con codice 99. Tratto da De Rinaldis)

```
481 // (inizio integrazione Spina)
482     vocabolario.inserisci("sforzo", 99);
483 // (fine integrazione Spina)
```

Righe 765-767 (Inserimento dell'azione di visualizzazione delle informazioni sulla distanza/sforzo, successivamente sostituito da "141" a "147".)

```
765 // (inizio integrazione Spina)
766 azioni.inserisci(990000, 147); // info su sforza fisico (lungo: 0000 (crescente), Verso: 0099, Distanza: 0000 (crescente))
767 // (fine integrazione Spina)
```

Righe 5786-5794 (Definizione metodo per la visualizzazione delle informazioni sulla distanza/sforzo, successivamente sostituito da "azione_141" ad "azione_147")

```
5786 // (inizio integrazione Spina)
5787 void Astro::azione_147() {
5788
5789     interfaccia.scrivi("*****calcolo sforzo*****");
5790     system("cls");
5791     sforzo.StampaPassi();
5792     interfaccia.scrivi("*****");
5793 }
5794 // (fine integrazione Spina)
```

Righe 6992-6996 (Chiamata del metodo per la visualizzazione delle informazioni sulla distanza/sforzo, successivamente sostituito da "141" a "147")

```
6992 // (inizio integrazione Spina)
6993 case 147:
6994     azione_147();
6995     break;
6996 // (fine integrazione Spina)
```

5.22.8 Grafo.h

Riga 1-369 (Integrato da De Rinaldis)

```

1 // (@Integratore: Alex Spina)
2
3 #ifndef _GRAFO_H
4 #define _GRAFO_H
5
6 #include <iostream>
7 #include <cstdlib>
8 #include "Lista.h"
9 #include "Adiacenza.h"
10 #include "Nodo_Grafo.h"
11 //definizione della classe grafo (orientata, etichettato e pesata)
12 //realizzazione tramite vettore (di lunghezza maxnodi) con liste (monodirezionali dinamiche) di adiacenza
13
14 template<class E, class P> class Grafo
15 {
16     // INIZIO modifica DE RINALDIS ANDREA
17     const int DEFAULT_MAX = 200;
18     // fine modifica DE RINALDIS ANDREA
19
20 public:
21
22     // dichiarazioni di tipo
23     //Tutte le fiche DE ROSE LUCA
24     typedef Nodo_Grafo nodo;
25     //Fine modifiche DE ROSE LUCA
26     typedef E tipoelem; //tipoelem (etichetta)
27     typedef P tipoarco; //tipoarco (peso)
28     typedef Adiacenza<Nodo_Grafo, tipoarco> adiacente; //tipo dell'elemento che farà parte della lista di adiacenza
29     //    adiacente = (int > nodo adiac | peso arco)
30
31     typedef struct //definizione dell'elemento del vettore
32     {
33         tipoelem etichetta; //etichetta del nodo di tipo tipoelem
34         bool esiste; //caso booleano che indica se il nodo fa parte del grafo
35         Lista<adiacente> adiac; //lista di adiacenze
36     } nodo_t;
37
38
39     // inizio modifiche DE RINALDIS ANDREA
40     Grafo(); // costruttore
41     // fine modifiche DE RINALDIS ANDREA
42
43     Grafo(unsigned int); // costruttore
44     ~Grafo(); // distruttore
45
46
47     unsigned int n_nodi();
48
49     //operatori di specifica
50
51     void creagrafo(unsigned int);
52     bool grafovuoto() const;
53     //Inizio modifiche DE ROSE LUCA
54     void insnodo(nodo&);
55     void insarco(nodo, nodo);
56     void cancnodo(nodo);
57     void cancarco(nodo, nodo);
58     Lista<nodo> adiacenti(nodo) const;
59     bool esistenodo(nodo) const;
60     bool esistearco(nodo, nodo) const;
61     void scrivinodo(tipoelem, nodo);
62     tipoelem legginodo(nodo) const;
63     void scriviarco(tipoarco, nodo, nodo);
64     tipoarco leggiarco(nodo, nodo);

```

```

65 //Fine modifica DE ROSE LUCA
66
67 private:
68     nodo_t* table; //contatore al vettore di liste (uso il contatore perché sarà allocato dinamicamente)
69     unsigned int maxnodi; //massimo numero dei nodi
70     unsigned int nelementi; //indica quanti nodi effettivamente ci sono nel grafo
71 //Inizio modifica DE ROSE LUCA
72     nodo primolibero() const; //inizializza la prima posizione libera del vettore (in modo da non avere un valore "strana")
73 //Fine modifica DE ROSE LUCA
74     void stampa(ostream&); //stampa per sovraccarico output
75
76     friend ostream& operator<<(ostream& os, Grafo<E,P>& g) //sovraccarico output
77     {
78         g.stampa(os);
79         return(os);
80     }
81
82 };
83
84 // inizia modifica DE RINALDIS ANDREA
85 template<class E, class P> Grafo<E,P>::Grafo()
86 {
87     creagrafo(DEFAULT_MAX);
88 }
89 // fine modifica DE RINALDIS ANDREA
90
91 //inizializza il vettore con zero
92 template<class E, class P> Grafo<E,P>::Grafo(unsigned int n) //sovraccarico secondario
93 {
94     creagrafo(n);
95 }
96
97 template<class E, class P> Grafo<E,P>::~Grafo() //destruttore
98 {
99     delete[] table;
100 }
101
102 template<class E, class P> void Grafo<E,P>::creagrafo(unsigned int n) //crea il grafo
103 {
104     table = new nodo_t[n]; //creo l'array di n elementi
105     maxnodi=n; //imposto ad n il numero massimo di nodi
106     nelementi=0; //inizializzo che ci sono 0 nodi
107     for (int i=0; i<n; i++) //inizializzo il vettore mettendo a false l'appartenenza di tutti nodi
108     {
109         table[i].esiste=false;
110     }
111 }
112
113 template<class E, class P> bool Grafo<E,P>::grafovuoto() const //restituisce true se il grafo è vuoto, false altrimenti
114 {
115     return (nelementi==0);
116 }
117
118 template<class E, class P> void Grafo<E,P>::insnodo(Nodo_Grafo &n) //inserisce il nodo che sarà identificato dall'indice n
119 {
120     //passaggio del parametro per indirizzo perché la variabile sarà modificata
121
122 //Inizio modifica DE ROSE LUCA
123     if (!esistenodo(n) || n.getNodo()==-1) && nelementi<maxnodi) // precondizione nodo non appartenente
124     {
125         n=primolibero(); //la posizione in cui metterà il nodo del vettore sarà la prima libera
126         table[n.getNodo()].esiste=true; //setto a true il suo campo esiste
127         nelementi++; //aumento il contatore di nodi del grafo
128     }
129 //Fine modifica DE ROSE LUCA
130 }
131
132 template<class E, class P> void Grafo<E,P>::insarco(Nodo_Grafo n,Nodo_Grafo m) //inserisce l'arco che esca dal nodo n ed entra in m
133 {
134     if (esistenodo(n) && esistenodo(m) && !esistearco(n,m))//precondizione nodi appartenenti e arco non esistente
135     {
136         Cella<adiacente*>* indice=table[n.getNodo()].adiac.primolista();
137         adiacente temp; //creo l'elemento da inserire
138         temp.scrivinodo(m); //setto l'adiacente
139
140 //Inizio modifica DE RINALDIS ANDREA
141         table[n.getNodo()].adiac.inslista(temp, indice); //inserisco il nodo m negli adiacenti di n (in prima posizione)
142 //Fine modifica DE RINALDIS ANDREA
143     }
144 }
145

```

```

146     template<class E, class P> void Grafo<E,P>::cancnodo(Nodo_Grafo n) //elimina il nodo n
147     {
148         if (esistenodo(n)) // 1)
149         {
150             //|
151             if (table[n].adiac.listavuota()) // 2)           //|
152             {
153                 //|
154                 bool libero=true;
155                 Nodo_Grafo i;
156                 i.setNodo(0);
157                 //int i=0;
158                 while (i.getNodo()<maxnodi && libero) //|
159                 //while (i<maxnodi && libero)           //|
160                 {
161                     //|--- Precondizione : nodo esistente 1), che non ha archi uscenti 2) né entranti 3)
162                     if (esistenodo(i)) //|
163                     {
164                         //|
165                         libero=!esistearco(i,n); //|
166                     }
167                     i.setNodo(i.getNodo()+1); //|
168                     //i++;
169                 } //|
170                 if (libero)// 3)-----+
171                 {
172                     table[n].esiste=false; //sairesso quel nodo non esiste più
173                     table[n].etichetta=NULL; //svuoto l'etichetta
174                     nelementi--; //e ho un nodo in meno nel grafo
175                 }
176             }
177         }
178     }
179 }
180
181     template<class E, class P> void Grafo<E,P>::cancarco(Nodo_Grafo n,Nodo_Grafo m) //elimina l'arco che esce da n ed entra in m
182     {
183         if (esistenodo(n) && esistenodo(m) && esistearco(n,m)) //precondizione nodi appartenenti e arco esistente
184         {
185             bool cancellato=false;
186             Cella<adiacente*>* indice=table[n].adiac.primolista();
187
188             //Inizio modifica DE ROSE LUCA
189             while (!table[n.getNodo()].adiac.finilista(indice) && !cancellato) //scandisco la lista finché non cancello l'elemento
190             {
191                 if (table[n.getNodo()].adiac.leggilista(indice).legginodo()==m) //se trovo l'elemento lo cancello
192                 {
193                     table[n.getNodo()].adiac.canclista(indice); //elimino l'elemento
194                     cancellato=true;
195                 }
196                 else indice=table[n].adiac.succlista(indice);
197             }
198             //Fine modifica DE ROSE LUCA
199         }
200     }

```

```

201 template<class E, class P> Lista<Nodo_Grafo> Grafo<E,P>::adiacenti(Nodo_Grafo n) const // restituisce la lista di adiacenti di n
202 {
203     Lista<Nodo_Grafo> lista; // lista da restituire
204     adiacente temp; // temporaneo
205     if (esistenodo(n)) // precondizione nodo appartiene al grafo
206     {
207         // Inizio modifica DE ROSE LUCA
208         Cell<adiacente>* indice=table[n.getNode()].adiac.primolista(); // indice di scansione della lista di adiacenti (lista di adiacenze)
209         Cell<Nodo_Grafo>* indice2=lista.primolista(); // indice di scansione della lista di adiacenze da aggiornare (lista di nodi)
210         while (!table[n.getNode()].adiac.finlista(indice)) // scandisco finché non trovo l'elemento o è finita
211         {
212             temp=table[n.getNode()].adiac.leggilista(indice); // lettura di adiacente
213             // Inizio modifica DE RINALDIS ANDREA
214             lista.inslista(temp.legginodo(), indice2); // inserisco (in coda) nella lista da restituire solo il riferimento al nodo adiacente (senza il
215             // fine modifica DE RINALDIS ANDREA
216             indice=table[n.getNode()].adiac.succlista(indice);
217             indice2=lista.succlista(indice2);
218         }
219         // Fine modifica DE ROSE LUCA
220     }
221     return(lista);
222 }
223
224
225
226
227 template<class E, class P> bool Grafo<E,P>::esistenodo(Nodo_Grafo n) const // restituisce true se il nodo appartiene al grafo, false altrimenti
228 {
229
230     // Inizio modifica DE ROSE LUCA
231     if (n.getNode()<maxnodi) return (table[n.getNode()].esiste);
232     // Fine modifica DE ROSE LUCA
233     else return false;
234 }
235
236 template<class E, class P> bool Grafo<E,P>::esistearco(Nodo_Grafo n, Nodo_Grafo m) const // restituisce true se il nodo n ha un arco uscente verso m, falso altrimenti
237 {
238     bool esiste=false;
239     if (esistenodo(n) && esistenodo(m)) // precondizione entrambi appartengono al grafo
240     {
241         // Inizio modifica DE ROSE LUCA
242         Cell<adiacente>* indice=table[n.getNode()].adiac.primolista();
243         while (!table[n.getNode()].adiac.finlista(indice) && !esiste) // scandisco la lista finché non trovo l'elemento o è finita
244         {
245             if (table[n.getNode()].adiac.leggilista(indice).legginodo().getNode() == m.getNode()) esiste=true;
246             indice=table[n.getNode()].adiac.succlista(indice);
247         }
248         // Fine modifica DE ROSE LUCA
249     }
250     return (esiste);
251 }
252
253 template<class E, class P> void Grafo<E,P>::scrivinodo(tipoelem val, Nodo_Grafo n) // scrive l'etichetta del nodo n
254 {
255     // Inizio modifica DE ROSE LUCA
256     table[n.getNode()].etichetta=val;
257     // Fine modifica DE ROSE LUCA
258 }
259
260 template<class E, class P> E Grafo<E,P>::legginodo(Nodo_Grafo n) const // restituisce l'etichetta del nodo n
261 {
262     tipoelem e;
263     if (esistenodo(n)) // precondizione nodo appartiene al grafo
264         // Inizio modifica DE ROSE LUCA
265         e=table[n.getNode()].etichetta;
266         // Fine modifica DE ROSE LUCA
267     return (e);
268 }
269
270 template<class E, class P> void Grafo<E,P>::scriviarco(tipoarco val, Nodo_Grafo n, Nodo_Grafo m) // scrive il peso dell'arco che va dal nodo n al nodo m
271 {
272     if (esistenodo(n) && esistenodo(m)) // precondizione nodi appartenenti al grafo (c'è anche arco appartenente ma non conviene altrimenti c'è una s
273     {
274         // un colpo e poi si faranno la seconda scansione per aggiornare
275         // Inizio modifica DE ROSE LUCA
276         bool aggiornato=false; // quindi si fa un'unica scansione in cui si ricerca e si aggiorna
277         adiacente temp(m, val);
278         Cell<adiacente>* indice=table[n.getNode()].adiac.primolista();
279         while (!table[n.getNode()].adiac.finlista(indice) && !aggiornato) // scandisco la lista finché non trovo l'elemento o è finita
280         {
281             if (table[n.getNode()].adiac.leggilista(indice).legginodo().getNode() == m.getNode())
282             {
283                 // Inizio modifica DE RINALDIS ANDREA
284                 table[n.getNode()].adiac.scrivilista(temp, indice);
285                 // fine modifica DE RINALDIS ANDREA
286                 aggiornato=true;
287             }
288             indice=table[n.getNode()].adiac.succlista(indice);
289         }
290         // Fine modifica DE ROSE LUCA
291     }
292 }

```

```

293     template<class E, class P> P Grafo<E,P>::leggiorco(Nodo_Grafo n,Nodo_Grafo m) //aggiornisce il peso dell'arco che va da n a m
294     {
295         tipoarco a;
296         if (esistenodo(n) && esistenodo(m)) //condizione per i connettori al grafo (c'è anche sotto aggiornamento ma non conviene altrimenti ciò una f
297         {
298             //se esiste e poi si farà la seconda scansione per la lettura
299             bool letto=false;
300             //Inizia modifica DE ROSE LUCA
301             Cella<adiacente>* indice=table[n.getNodo()].adiac.primolista();
302             while (!table[n.getNodo()].adiac.finelist(indice) && !letto) //aggiorna la lista finché non trova l'elemento o è finita
303             {
304                 if ((table[n.getNodo()].adiac.leggilista(indice).legginodo()).getNodo() == m.getNodo())
305                 {
306                     a=table[n.getNodo()].adiac.leggilista(indice).leggipeso();
307                     letto=true;
308                 }
309             }
310             indice=table[n.getNodo()].adiac.succlista(indice);
311         }
312         //Fine modifica DE ROSE LUCA
313     }
314     return(a);
315 }
316
317
318 //operatori ausiliari
319
320
321     template<class E, class P> Nodo_Grafo Grafo<E,P>::primolibero() const //aggiornisce la prima posizione del vettore libero
322     {
323         Nodo_Grafo i;
324         //Inizia modifica DE ROSE LUCA
325         i.setNodo(-1);
326         bool libero=false;
327         while (!libero) //sorpre il vettore finché non trova una posizione libera
328         {
329             //il controllo è solo su libero perché se già mi fermerò per forza poiché nessun metodo viene chiamato solo se il vettore non è pieno
330             i.setNodo(i.getNodo()+1);
331             libero=!table[i.getNodo()].esiste();
332         }
333         //Fine modifica DE ROSE LUCA
334         return (i);
335     }
336
337     template<class E, class P> void Grafo<E,P>::stampa(ostream& os) //stampa il grafo per overload output
338     {
339         for (int i=0; i<maxnodi; i++) //scatta tutto il vettore
340         {
341             if (table[i].esiste) //per ogni nodo esistente ne stampo l'etichetta e la lista di adiacenti
342             {
343                 os<<table[i].etichetta<<"-> <";
344                 Cella<adiacente>* indice=table[i].adiac.primolista();
345                 adiacente temp;
346                 while (!table[i].adiac.finelist(indice))
347                 {
348                     //Inizia modifica DE ROSE LUCA
349                     temp=table[i].adiac.leggilista(indice);
350                     //Fine modifica DE ROSE LUCA
351
352                     os<<legginodo(temp.legginodo())<<"#"<<temp.leggipeso();
353                     indice=table[i].adiac.succlista(indice);
354                     if (!table[i].adiac.finelist(indice)) os<<, ";
355                 }
356                 os<<">"<<endl;
357             }
358         }
359     }
360
361     template<class E, class P> unsigned int Grafo<E,P>::n_nodi()
362     {
363         return nelementi;
364     }
365
366
367 #endif
368
369

```

5.22.9 Nodo_Grafo.h

Righe 1-20 (Integrato da De Rinaldis)

```

1 // (@Integratore: Alex Spina)
2
3 #ifndef NODO_GRAFO_H
4 #define NODO_GRAFO_H
5
6 class Nodo_Grafo{
7 public:
8     Nodo_Grafo();
9     int getNode();
10    void setNode(int);
11    bool operator==(Nodo_Grafo);
12
13 private:
14     int nodo;
15 };
16
17
18
19 #endif // NODO_GRAFO_H
20

```

5.22.10 Nodo_Grafo.cpp

Righe 1-18 (Integrato da De Rinaldis)

```

1 // (@Integratore: Alex Spina)
2
3 #include "Nodo_Grafo.h"
4 Nodo_Grafo::Nodo_Grafo(){
5     nodo=-1;
6 }
7 int Nodo_Grafo::getNode(){
8     return nodo;
9 }
10 void Nodo_Grafo::setNode(int n){
11     nodo=n;
12 }
13 bool Nodo_Grafo::operator==(Nodo_Grafo x)
14 {
15     return( this->nodo==x.nodo);
16 }
17
18

```

5.23 FABBRICA

5.23.1 Mappa.nav

44,Nella fabbrica,000000350000, via 6,2,2

35,Nella sala giochi,000044150000, via 6,2,2

5.23.2 Astro.cpp

5.23.2.1 Vocabolario

```
463 //INIZIO MODIFICHE PIERLUIGI BEMPORTATO  
464 vocabolario.inserisci("fabbricante",108);  
465 //FINE MODIFICHE PIERLUIGI BEMPORTATO
```

5.23.2.2 Oggetti

```
1038 //INIZIO MODIFICHE PIERLUIGI BEMPORTATO  
1039 oggetti.inserisci(Oggetto("un fabbricante", 108, -44));  
1040 //FINE MODIFICHE PIERLUIGI BEMPORTATO
```

5.23.2.3 Azione 148

```
790 //INIZIO MODIFICHE PIERLUIGI BEMPORTATO  
791 azioni.inserisci(4400300108,148); //parla con fabbricante  
792 //FINE MODIFICHE PIERLUIGI BEMPORTATO
```


5.23.3 Fabbrica.h

```
1 #ifndef FABBRICA_H
2 #define FABBRICA_H
3
4 using namespace std;
5
6 #include <iostream>
7 #include <ctime>
8 #include <stdlib.h>
9 #include <string>
10
11 #include "Coda.h"
12
13 class Fabbrica
14 {
15 public:
16     Fabbrica();
17     virtual ~Fabbrica();
18     float scelta(int);
19
20 protected:
21     void crea_costruzioni();
22     void costruisci(Coda<string>, int);
23
24 };
25
26 #endif
```

5.23.4 Fabbrica.cpp

5.23.4.1 Variabili globali, costruttore e distruttore

```
1 #include "fabbrica.h"
2
3     Coda<string> collana;
4     Coda<string> tavolo;
5     Coda<string> porta;
6     Coda<string> computer;
7     Coda<string> missile;
8
9     Fabbrica::Fabbrica() //costruttore in cui viene richiamata la funzione crea_costruzioni
10 {
11     crea_costruzioni();
12 }
13
14 Fabbrica::~Fabbrica() {}
```

```
16 void Fabbrica::crea_costruzioni() //funzione che crea le code necessarie alle costruzioni
17 {
18     //costruzione coda collana
19     collana.creacoda();
20     collana.incoda("Collana");//nome coda
21     collana.incoda("filo");
22     collana.incoda("perline");
23     collana.incoda("gancio");
24
25     //costruzione coda tavolo
26     tavolo.creacoda();
27     tavolo.incoda("Tavolo");//nome coda
28     tavolo.incoda("piano in legno");
29     tavolo.incoda("gambe");
30     tavolo.incoda("giuntura");
31     tavolo.incoda("viti");
32
33     //costruzione coda porta
34     porta.creacoda();
35     porta.incoda("Porta in metallo");//nome coda
36     porta.incoda("telaio");
37     porta.incoda("pannello in metallo");
38     porta.incoda("cerniera");
39     porta.incoda("serratura");
40     porta.incoda("maniglia");
```

5.23.4.2 Funzione crea_costruzioni

```
41 //costruzione coda computer
42 computer.creacoda();
43 computer.incoda("Computer");//nome coda
44 computer.incoda("scheda madre");
45 computer.incoda("processore");
46 computer.incoda("scheda video");
47 computer.incoda("ventola");
48 computer.incoda("case");
49 computer.incoda("periferiche");
50
51 //costruzione coda missile
52 missile.creacoda();
53 missile.incoda("Missile");//nome coda
54 missile.incoda("motore");
55 missile.incoda("tubo di scarico");
56 missile.incoda("sensore infrarossi");
57 missile.incoda("giroscopio");
58 missile.incoda("sistema di volo");
59 missile.incoda("detonatore");
60 missile.incoda("carica esplosiva");
61 missile.incoda("scocca");
62 missile.incoda("giunture");
63 missile.incoda("alette");
64
65
66
67 }
68 }
```

5.23.4.3 Funzione costruisci

5.23.4.4 Funzione scelta

```
134 //funzione che prende in input la scelta dell'utente e seleziona la coda
135 //restituisce il guadagno per aggiornare il portafoglio in Astro.cpp
136 float Fabbrica::scelta(int s)
137 {
138     float guadagno = 0;
139     switch(s)
140     {
141         case 1:
142             Fabbrica::costruisci(collana, 4);
143             guadagno = 2;
144
145             return guadagno;
146
147         break;
148
149         case 2:
150             Fabbrica::costruisci(tavolo, 5);
151             guadagno = 10;
152
153             return guadagno;
154
155         break;
156
157         case 3:
158             Fabbrica::costruisci(porta, 6);
159             guadagno = 20;
160
161             return guadagno;
162
163         break;
164
165         case 4:
166             Fabbrica::costruisci(computer, 7);
167             guadagno = 50;
168
169             return guadagno;
170
171         break;
172
173         case 5:
174             Fabbrica::costruisci(missile, 11);
175             guadagno = 100;
176
177             return guadagno;
178
179         break;
180
181         default:
182             return guadagno;
183         break;
184     }
185 }
186
187 }
```

5.24 FUNZIONE TELETRASPORTA

Per realizzare la funzionalità sono stati modificati i seguenti file:

- Astro.h
- Astro.cpp
- Gioco.h

a. Astro.h

La modifica di questo file riguarda la dichiarazione dell'azione che permetterà di teletrasportarsi e che sarà implementata in seguito, inoltre ad essa verrà assegnato un codice univoco (149).

Dalla riga di codice 290 alla riga di codice 292:

```
290 //inizio modifiche Bagordo Simone
291 void azione_149();
292 //fine modifiche Bagordo Simone
293 };
294
```

b. Gioco.h

La modifica del file riguarda la dichiarazione della Pila teletrasporta che verrà utilizzata come struttura per implementare la funzionalità di teletrasporto.

Dalla riga di codice 238 alla riga di codice 240:

```
238 //inizio modifiche Bagordo Simone
239 Pila<int> teletrasporta;
240 //fine modifiche Bagordo Simone
241
```

La pila verrà utilizzata per salvare tutti i luoghi, dal più vecchio al più recente, nei quali il giocatore ha utilizzato il comando “teletrasporta”, in modo tale che, nel caso in cui il giocatore volesse sfruttare la possibilità di andare indietro (digitando appunto “indietro”), verrà trasportato immediatamente nell'ultimo luogo in cui ha utilizzato tale comando.

c. Astro.cpp

La modifica di questo file riguarda:

- Vocabolario
- Azioni
- Procedure

Viene definito all'interno del vocabolario del gioco il verbo “teletrasporta” con codice identificativo 135

Dalla riga di codice 487 alla riga di codice 489:

```
487 //inizio modifiche Bagordo Simone
488 vocabolario.inserisci("teletrasporta", 135);
489 //fine modifiche Bagordo Simone
```

Viene inserita l'azione che permetterà il teletrasporto definito nei paragrafi precedenti

Dalla riga di codice 798 alla riga di codice 800:

```
798 //inizio modifiche Simone Bagordo
799 azioni.inserisci(1350000, 149);
800 //fine modifiche Simone Bagordo
```

Viene implementata l'azione “teletrasporta”

Dalla riga di codice 6595 alla riga di codice 6644:

```
6595 //AZIONE TELETRASPORTA
6596 //inizio modifiche Simone Bagordo
6597 void Astra::azione_149(){
6598     string luogo_teletrasporta;
6599     int luogo_teletrasporta_int;
6600
6601     do{
6602         luogo_teletrasporta = interfaccia.leggi_stringa("In quale luogo desideri essere teletrasportato?");
6603
6604         for (int i = 0; i < luogo_teletrasporta.length(); i++)
6605             if (isdigit(luogo_teletrasporta[i])){
6606                 luogo_teletrasporta_int = stoi( luogo_teletrasporta );
6607             }
6608
6609         if (luogo_teletrasporta != "indietro" && ( luogo_teletrasporta_int < 1 || luogo_teletrasporta_int > 42)){
6610             cout << "Non ho capito!\n";
6611         }
6612
6613     }while(luogo_teletrasporta != "indietro" && ( luogo_teletrasporta_int < 1 || luogo_teletrasporta_int > 44));
6614
6615
6616     if (luogo_teletrasporta != "indietro"){
6617
6618         if(luogo_attuale == luogo_teletrasporta_int){
6619             cout << "Ti trovi già nel luogo specificato!\n";
6620         }
6621         else{
6622
6623             if ( teletrasporta.pilavuota(){ //Initial place where teleportation was used
6624                 teletrasporta.inpila(luogo_attuale);
6625             }
6626
6627             luogo_attuale = luogo_teletrasporta_int;
6628             teletrasporta.inpila(luogo_teletrasporta_int);
6629         }
```

```

6630 }
6631 else if ( luogo_teletrasporta == "indietro" && !teletrasporta.pilavuota()){
6632     teletrasporta.fuoripila();
6633     if (!teletrasporta.pilavuota() ){
6634         luogo_attuale = teletrasporta.leggipila();
6635     }
6636     else{
6637         cout << "Ti trovi al punto di partenza del teletrasporto!\n";
6638     }
6639 }
6640 else{
6641     cout << "Ti trovi al punto di partenza del teletrasporto!\n";
6642 }
6643 }
6644 //fine modifiche Simone Bagordo

```

Qui sono stati definiti 3 messaggi di errore opportuni per i seguenti casi:

- Luogo non esistente sulla mappa: “non ho capito!”
- Il giocatore si trova già nel luogo verso cui ha richiesto di essere teletrasportato: “ti trovi già nel luogo specificato!”
- L’utente, dopo aver richiesto di essere teletrasportato indietro, si trova già nel luogo di partenza: “ti trovi già al punto di partenza del teletrasporto!”

Viene infine inserito nello switch case il richiamo all’azione implementata

Dalla riga di codice 7134 alla riga di codice 7138:

```

7134 //INIZIO MODIFICHE Bagordo Simone
7135 case 149:
7136     azione_149();
7137 break;
7138 //FINE MODIFICHE Bagordo Simone

```

5.25 CASSAFORTE E CASSETTA DI SICUREZZA MALMESSA, CORREZIONE VERBO “USA”, “PARLA”, INDICI MAPPE E OGGETTI, TESSERA

a. Mappa.nav

Modifica della mappa 15 per il nuovo percorso e aggiunta della mappa 54:

```

15,In una banca,235435140000,via 6,2,2
54,In una cassaforte,150000000000,via 6,2,3

```

Aggiunta di due nuove righe:

```

54,15,via 6,nord,1,1,1,1
15,54,via 6,sud,1,1,1,1

```

Visto che il numero di mappe, considerando la cassaforte e la caserma, supera il numero di mappe da caricare, è stata modificata anche la prima riga contenente il numero rappresentante questa informazione.

b. Descrizioni (Cartella)

Aggiunta del file "54.txt" con questo contenuto

```
|ancora una volta nella cassaforte  
nella cassaforte  
di nuovo nella cassaforte  
nuovamente nella cassaforte  
ancora nella cassaforte
```

c. Astro.cpp

i. Inclusione Cassetta_Sicurezza.h

```
#include "Cassetta_Sicurezza.h" // Implementazione di Samuele Pesce dal progetto di Roberto Carlucci.
```

ii. Inserimento azioni nel dizionario

```
//Implementazione di Samuele Pesce per integrare il di Roberto Carlucci.  
azioni.inserisci(4700099999,150); // deposita oggetto  
azioni.inserisci(4700350041,151); // controlla cassetta  
azioni.inserisci(4700429999,152); // ritira oggetto  
//Fine modifica di Samuele Pesce
```

iii. Inserimento oggetti nel dizionario

```
//Implementazione di Samuele Pesce per integrare il di Roberto Carlucci.  
oggetti.inserisci(Oggetto("una cassetta di sicurezza malmessa", 120, -47));  
//Fine modifica di Samuele Pesce
```

iv. Implementazione azioni

```
//Implementazione di Samuele Pesce per integrare il di Roberto Carlucci.  
void Astro::azione_150() { cassetta.deposita_oggetto(comando_oggetto, interfaccia, oggetti); }  
void Astro::azione_151() { cassetta.mostra_contenuto(interfaccia); }  
void Astro::azione_152() { cassetta.ritira_oggetto(comando_oggetto, interfaccia, oggetti); }  
//Fine modifica Samuele Pesce
```

v. Inserimento azioni in astro:bool_esegui_specifiche()

```
//Implementazione di Samuele Pesce per integrare il di Roberto Carlucci.  
case 150: azione_150(); break;  
case 151: azione_151(); break;  
case 152: azione_152(); break;  
//Fine modifica Samuele Pesce
```

vi. Modifica del teletrasporto inserito dal collega Bagordo per supportare le nuove mappe

```
do{  
    luogo_teletrasporta = interfaccia.leggi_stringa("In quale luogo desideri essere teletrasportato?");  
    for (int i = 0; i < luogo_teletrasporta.length(); i++)  
        if (isdigit(luogo_teletrasporta[i])) {  
            luogo_teletrasporta_int = stoi(luogo_teletrasporta);  
        }  
  
        if (luogo_teletrasporta != "indietro" && ( luogo_teletrasporta_int < 1 || luogo_teletrasporta_int > 47)){ //Modifica Samuele Pesce  
            cout << "Non ho capito!\n";  
        }  
}  
  
while(luogo_teletrasporta != "indietro" && ( luogo_teletrasporta_int < 1 || luogo_teletrasporta_int > 47)); // Modifica di Samuele Pesce
```

vii. Correzione delle azioni che sfruttano il verbo "usa"

Durante i testo ho notato che nel luogo "Palestra" non era possibile interagire con niente e, dopo aver aver indagato, ho corretto il verbo dell'azione "usa" che era stato modificato da 29 a 80, senza però poi implementare questa modifica nelle azioni.

```
//INIZIO modifiche DAVIDE MANTELLINI -- Palagiano Marcelllo: modifica codice luogo da 35 a 34  
azioni.inserisci(3400800057, 120); //usa panca -----+ -- Inizio modifiche Samuele Pesce  
azioni.inserisci(3400800047, 121); //usa tapis |  
azioni.inserisci(3400800099, 122); //usa terminale | --- Modifica di Samuele Pesce per aggiustare il nuovo id  
azioni.inserisci(3400100099, 122); //guarda terminale | del verbo guarda: 80.  
azioni.inserisci(3400800084, 123); //usa schede |  
azioni.inserisci(3400800097, 124); //usa macchinetta -----+ -- Fine modifiche Samuele Pesce  
//FINE modifiche DAVIDE MANTELLINI
```

Le seguenti modifiche derivano dalla stessa motivazione:

```
//azione che permette di usare il tesserino  
azioni.inserisci(800800031, 59); //Modifica Pesce Samuele correzione id "Usa": 80.
```

```
azioni.inserisci(1900809999, 69); //Azione per usare il terminale nel pronto soccorso //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(2600809999, 91); //uso del computer //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(3200800056, 109); //Interagisci con il Jukebox nell'auditorium //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(3200100088, 110); //Guarda gli strumenti musicali  
azioni.inserisci(3200720088, 111); //Usa gli strumenti musicali  
azioni.inserisci(3200800070, 112); //Interagisci con il pannello delle luci //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(3300800080, 115); //usa biglietteria" //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(4900800048, 118); //Se si usa il teletrasporto nell'astronave aliena -- Palagiano Marcello: modifica codice luogo da 40 a 48 //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(1100800049, 118); //Se si usa la pietra canalizzatrice nel luogo 11 "Poppa dell'astronave" //Modifica Pesce Samuele correzione id "Usa": 80.  
  
azioni.inserisci(4300800104,145); //usa elaboratore //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(4300800104,145); //usa elaboratore bonus //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(4300800104,145); //usa elaboratore //Modifica Pesce Samuele correzione id "Usa": 80.  
azioni.inserisci(4300800104,145); //usa elaboratore bonus //Modifica Pesce Samuele correzione id "Usa": 80.
```

viii. Correzione conflitto id ‘Tessera’

Nei test è risultato impossibile prendere la tessera sanitaria nella propria cabina. Dopo un'indagine, è risultato evidente un conflitto tra gli id 84 e 31. Visto che poi l'id 84 risultava utilizzato anche dalle schede di allenamento della palestra, si è pensato di utilizzare l'id 127 per risolvere il problema a monte. Fatto questo, si è anche corretta la funzione nel Pronto Soccorso per renderla funzionante col nuovo id, anziché di farla funzionare come prima con l'indice dell'oggetto nel vettore.

```

vocabolario.inserisci("tessera", 127); //Modifica di Samuele Pesce per risolvere eventuali conflitti di ID
//Modifica di Samuele Pesce per permettere di far funzionare usando l'id della tessera e non l'indice.
oggetti.inserisci(Oggetto("una tessera sanitaria",127, 6)); //Modifica Pesce Samuele

//Modifica di Samuele Pesce per permettere di far funzionare usando l'id del gettone e non l'indice.
if(oggetti.get_oggetto(oggetti.get_oggetto_indice_by_codice(127)).get_luogo() == 0) //Se hai la tessera nell'inventario
{
    interfaccia.scrivi("----- TESSERA SANITARIA RICONOSCIUTA! -----");
    interfaccia.scrivi("----- DIGITA 0 PER CURARTI! -----");
    interfaccia.scrivi("----- DIGITA 1 PER SPEGNERE IL TERMINALE! -----");
} //Modifica di Samuele Pesce per permettere di far funzionare usando l'id del gettone e non l'indice.
else if(oggetti.get_oggetto(oggetti.get_oggetto_indice_by_codice(86)).get_luogo() == 0) //Se hai il gettone
{

```

ix. Correzione problemi coi personaggi

Nei test è risultato impossibile parlare coi personaggi. Dopo un'indagine, si è notato che il verbo "Parla" era sia considerato sinonimo di "Consulta", entrambi ID 30, che serve a far funzionare rubrica e guida, sia inserito come doppione con id 39. Per risolvere il problema, si è rimossa la dichiarazione con l'id 30, lasciando solo quello con 39, e si sono modificate le funzioni che riguardavano l'azione 30 intesa come "Parla" e non come "Consulta".

```
//vocabolario.inserisci("parla", 30);      Rimozione Samuele Pesce per conflitto con id 39
azioni.inserisci(2000390098, 73); //Azione per parlare al sacerdote
```

d. Oggetti.h

Visto che l'inserimento del nuovo oggetto andava a superare il limite massimo di oggetti nel gioco, questo file è stato modificato per aumentare questo limite e portarlo da 150 a 200.

```
#define MAXOGGETTI 200
```

La funzione exit, usata in seguito per fermare l'esecuzione del programma, richiede l'include del file `<stdlib.h>`, che era già presente e quindi non è stato necessario includere altro.

2. Oggetti cpp

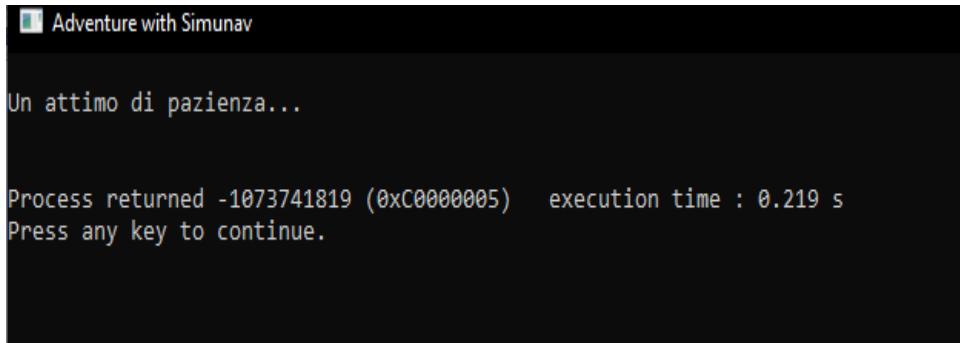
Oggetti.cpp
Aggiunta di un controllo e di un messaggio di errore nel caso in cui siano stati inseriti troppi oggetti.

```

void Oggetti::inserisci(Oggetto oggetto)
{
    //Modifica di Samuele Pesce per aiutare a capire se si è superato il numero massimo di oggetti.
    if( get_n_oggetti() == MAXOGGETTI-1){
        cout<<"Hai inserito troppi oggetti nel sistema. Il massimo attuale e' "<<MAXOGGETTI<<" modifica la costante simbolica MAXOGGETTI nel file oggetti.h"<<endl;
        exit(EXIT_FAILURE);
    } else oggetti[+fog] = oggetto;
    //Fine modifica Samuele Pesce
}

```

Prima della correzione:

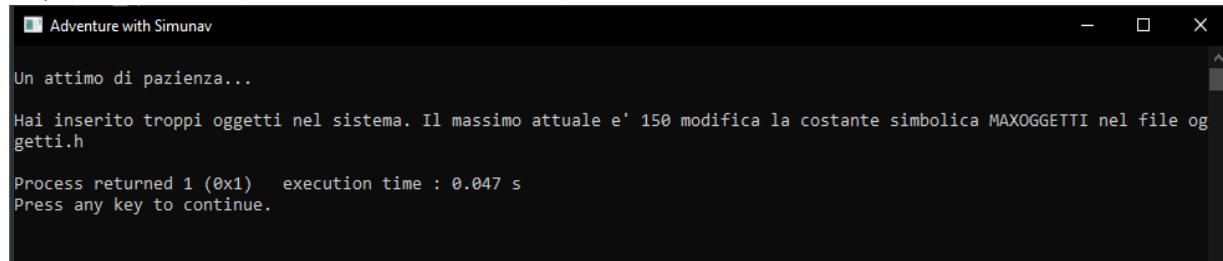


```
Adventure with Simunav

Un attimo di pazienza...

Process returned -1073741819 (0xC0000005) execution time : 0.219 s
Press any key to continue.
```

Dopo la correzione (senza aumentare il limite)



```
Adventure with Simunav

Un attimo di pazienza...

Hai inserito troppi oggetti nel sistema. Il massimo attuale e' 150 modifica la costante simbolica MAXOGGETTI nel file oggetti.h

Process returned 1 (0x1) execution time : 0.047 s
Press any key to continue.
```

f. Astro.h

```
//Inizio implementazione di Samuele Pesce dal progetto di Roberto Carlucci.
void azione_150(); // deposita oggetto
void azione_151(); // controlla cassetta
void azione_152(); // ritira oggetto
//Fine modifica Samuele Pesce
```

g. Gioco.cpp

Implementazione dello sfalsamento degli indici nella mappa normale, come spiegato in Progettazione

```
a = b;
if((autobus.get_inUso() == false) && (automobile.get_inUso() == false)) {
    interfaccia.scrivi("- Impieghi piu' tempo muovendoti a piedi");
    tempo = tempo - 3;
}
else if(a == 48) a -= 3; //Sfalsamento necessario per la mappa con id 48
else if(a > 53) a -= 7; //Sfalsamento necessario per le mappe superiori alla 53. -| Modifica Samuele Pesce per implementare Cassaforte dal progetto di Roberto Carlucci
```

h. Gioco.h

Inclusione cassetta_sicurezza.h e creazione nuovo oggetto "cassetta" di classe Cassetta.

```
28 #include "Banca.h"      //MODIFICA D-R(D'Orsi):Negozio + Banca
29
30 #include "Cassetta_Sicurezza.h" // Implementazione di Samuele Pesce dal progetto di Roberto Carlucci.
31
32 #include "Portafoglio.h"    //MODIFICA D-R(Pisani):Portafoglio + Carta di Credito
...
//Inizio implementazione di Samuele Pesce dal progetto di Roberto Carlucci.
Cassetta cassetta;
//Fine modifica Samuele Pesce

// metodi
```

i. Cassetta_Sicurezza.h

```
//Rimozione della struct con un solo campo

class Cassetta{
public:
    Cassetta();

    void deposita_oggetto(int, Interfaccia&, Oggetti&);
    void mostra_contenuto(Interfaccia&);
    void ritira_oggetto(int, Interfaccia&, Oggetti&);

protected:
    const int maxInt = 10;           //--|
    Oggetto* comparti;            //---+Modifica Samuele Pesce
    int contenuto=0;
};
```

j. Cassetta_Sicurezza.cpp

```
#include "Cassetta_Sicurezza.h"
//Fatto da Roberto Carlucci reso strutturato da Samuele Pesce

Cassetta::Cassetta()
{
    comparti = new Oggetto[maxInt];
}

void Cassetta::deposita_oggetto(int codice, Interfaccia& interfaccia, Oggetti& inventario) {
    if(contenuto<maxInt) {
        if(inventario.get_luogo(inventario.get_oggetto_indice_by_codice(codice))==0) {
            comparti[contenuto]=inventario.get_oggetto(inventario.get_oggetto_indice_by_codice(codice));
            inventario.set_luogo(inventario.get_oggetto_indice_by_codice(codice),99);
            interfaccia.scrivi("Fatto\n");
            contenuto++;
        } else {
            interfaccia.scrivi("Non ce l'hai\n");
        }
    } else{
        interfaccia.scrivi("La cassetta e' piena purtroppo\n");
    }
}

void Cassetta::mostra_contenuto(Interfaccia& interfaccia) {
    if(contenuto!=0) {
        interfaccia.scrivi("All'interno della cassetta vedo: ");
        for(int i=0;i<contenuto;i++) {
            interfaccia.scrivi_parziale(" - ");
            interfaccia.scrivi(comparti[i].get_nome());
        }
    } else interfaccia.scrivi("La cassetta di sicurezza e' vuota");
}

void Cassetta::ritira_oggetto(int codice, Interfaccia& interfaccia, Oggetti& inventario)
{
    bool trovato = false;
    if(contenuto!=0) {
        int i = 0;
        while( i < contenuto && trovato == false ){
            if(codice == comparti[i].get_codice()){
                trovato = true;
            }
        }
    }
}
```

```

        inventario.set_luogo(inventario.get Oggetto indice_by codice(codice), 0);
        for(int j=i;j<contenuto-1;j++) comparti[j]=comparti[j+1];
        contenuto--;
        interfaccia.scrivi("Fatto\n");
    }
    i++;
}
}
if(!trovato) interfaccia.scrivi("L' oggetto richiesto non e' disponibile");
}

```

5.26 VERBO “RITORNA” E VERBO “RESOCOMTO”

Per realizzare la funzionalità sono stati modificati i seguenti file:

- Astro.cpp
- Gioco.h
- Gioco.cpp

Astro.cpp

La modifica di questo file riguarda la dichiarazione dell’azione che permetterà di ritornare e visualizzare il resoconto, che saranno poi implementate.

Per “ritorna” è stato necessario modificare anche alcuni comandi esistenti per evitare conflitti

```

476     vocabolario.inserisci("ritorna", 72);
477     azioni.inserisci(720000, 9); // torna indietro di un luogo
68      vocabolario.inserisci("annulla", 72);
98      vocabolario.inserisci("indietro", 72);
153     vocabolario.inserisci("torna", 72);
381     vocabolario.inserisci("suona", 222);
315     vocabolario.inserisci("spegni", 223);
688     azioni.inserisci(2502230080, 90);
719     azioni.inserisci(3202220080, 111);

```

Per “resoconto” è stato sufficiente aggiungere il vocabolo e la relativa azione

```

479 //INIZIO MODIFICHE LEANDRA PALEMBURGI
480     vocabolario.inserisci("resoconto", 12);
481     azioni.inserisci(120000, 12); //azione resoconto
482 //FINE MODIFICHE LEANDRA PALEMBURGI

```

Gioco.h

In questo file vengono integrati

La pila percorso, che sarà poi usata dalla funzionalità “ritorna”

```

95     protected:
96
97     Pila <string> percorso;//MODIFICA SPECCHIA ROBERTO

```

Il metodo ritorna

```
319 //INIZIO MODIFICHE SPECCHIA ROBERTO  
320     void ritorna(Pila <stato_comando>* sc, Mappa &M);  
321 //FINE MODIFICHE SPECCHIA ROBERTO
```

E il metodo resoconto

```
322 //INIZIO MODIFICHE LEANDRA PALEMBURGI  
323     void resoconto();  
324 //FINE MODIFICHE LEANDRA PALEMBURGI
```

Gioco.cpp

In questo file viene integrata la funzionalità “ritorna”, con relativa modifica per evitare un errore della funzione assert all’avvio del gioco, nel caso in cui si digiti il comando “w” non precedentemente previsto

```
1580 //INIZIO MODIFICHE SPECCHIA ROBERTO  
1581 void Gioco::ritorna(Pila <stato_comando>* sc, Mappa &M)  
1582 {  
1583  
1584     if (!percorso.pilavuota())  
1585     {  
1586  
1587         string com= percorso.leggipila();  
1588  
1589         if( com.compare("n") == 0 ||  
1590             com.compare("nord") == 0 ||  
1591             com.compare("N") == 0 ||  
1592             com.compare("NORD") == 0 ||  
1593             com.compare("Nord") == 0 )  
1594         {  
1595             com="s";  
1596         }
```

```

1597         else if( com.compare("o") == 0 ||  

1598             com.compare("ovest") == 0 ||  

1599             com.compare("O") == 0 ||  

1600             com.compare("OVEST") == 0 ||  

1601             com.compare("Ovest") == 0 )  

1602     {  

1603         com="e";  

1604     }  

1605 //INIZIO MODIFICA LEANDRA PALEMBURGI  

1606     else if( com.compare("w") == 0 ||  

1607             com.compare("west") == 0 ||  

1608             com.compare("W") == 0 ||  

1609             com.compare("WEST") == 0 ||  

1610             com.compare("West") == 0 )  

1611     {  

1612         com="w";  

1613     }  

1614 //FINE MODIFICA LEANDRA PALEMBURGI  

1615     else if( com.compare("e") == 0 ||  

1616             com.compare("est") == 0 ||  

1617             com.compare("E") == 0 ||  

1618             com.compare("EST") == 0 ||  

1619             com.compare("Est") == 0 )  

1620     {  

1621         com="o";  

1622     }  

1623  

1624 /* TEST STAMPA con pila di appoggio (app)  

1625     cout << com << "<- Ultimo invertito \n\n" << endl;  

1626     Pila <string> app;  

1627     while (!percorso.pilavuota())  

1628     {  

1629         cout << percorso.leggipila() << "-Pila\n" << endl;  

1630         app.inpila(percorso.leggipila());  

1631         percorso.fuoripila();  

1632     }  

1633     while (!app.pilavuota())  

1634     {  

1635         percorso.inpila(app.leggipila());  

1636         app.fuoripila();  

1637     }  

1638 */  

1639     percorso.fuoripila();  

1640     interfaccia.scrivi("Hai scelto di tornare tornare di un luogo indietro");  

1641     interfaccia.scrivi("\n\n");  

1642 // string stato_comando1 = p->leggipila().get_comando();  

1643     Stato stato= sc->leggipila().get_stato();  

1644  

1645     interpreta(com,M,sc,stato,false);  

1646     storia_gioco.insStoria("ritorna", stringa_risposta);  

1647 }  

1648  

1649 else  

1650 {  

1651     interfaccia.scrivi("Non puoi tornare indietro, non hai ancora iniziato nessun percorso");  

1652     interfaccia.scrivi("\n\n");  

1653 }  

1654 //FINE MODIFICA SPECCHIA ROBERTO

```

integrata la modifica di Specchia per inserire i comandi nella pila "percorso"

```
2314 //INIZIO MODIFICHE - SPECCHIA ROBERTO
2315     if (
2316         comando.compare("n") == 0 ||
2317         comando.compare("nord") == 0 ||
2318         comando.compare("N") == 0 ||
2319         comando.compare("NORD") == 0 ||
2320         comando.compare("Nord") == 0 ||
2321         comando.compare("o") == 0 ||
2322         comando.compare("ovest") == 0 ||
2323         comando.compare("O") == 0 ||
2324         comando.compare("OVEST") == 0 ||
2325         comando.compare("Ovest") == 0 ||
2326         comando.compare("e") == 0 ||
2327         comando.compare("est") == 0 ||
2328         comando.compare("E") == 0 ||
2329         comando.compare("EST") == 0 ||
2330         comando.compare("Est") == 0 ||
2331         comando.compare("s") == 0 ||
2332         comando.compare("sud") == 0 ||
2333         comando.compare("S") == 0 ||
2334         comando.compare("SUD") == 0 ||
2335         comando.compare("Sud") == 0 ||
2336         comando.compare("sali") == 0 ||
2337         comando.compare("SALI") == 0 ||
2338         comando.compare("Sali") == 0 ||
2339         comando.compare("scendi") == 0 ||
2340         comando.compare("SCENDI") == 0 ||
2341         comando.compare("Scendi") == 0 )
2342     {
2343         percorso.inpila(comando);
2344     }
```

A cui viene apportata la seguente modifica per la risoluzione del problema relativo al comando "w" riscontrato all'inizio del gioco

```
2326 .....
2327 //INIZIO MODIFICHE LEANDRA PALEMBURGI
2328 comando.compare("w") == 0 ||
2329 comando.compare("west") == 0 ||
2330 comando.compare("W") == 0 ||
2331 comando.compare("WEST") == 0 ||
2332 //FINE MODIFICHE LEANDRA PALEMBURGI
```

Integrazione della modifica della funzione "direzioni" del collega Specchia

```
1062     bool Gioco::direzioni(Mappa &M) {
1063         //INIZIO modifiche DELLA FOLGORE GRAZIA
1064         bool cambia_dir = true;
1065         int a = mappa.luogo_adiacente(luogo_attuale,cod_parola);
1066         if (a == 0){
1067             interfaccia.scrivi("- Di lì! non puoi andare");
1068             stringa_risposta = "di lì! non sei potuto andare.";
1069             storia_gioco.insStoria(stringa_comando , stringa_risposta);
1070             cambia_dir = false; //PANNO
1071             percorso.fuoripila(); //MODIFICA SPECCHIA ROBERTO
```

Integrazione della modifica alla funzione "esegui" per inserire la chiamata all'azione "ritorna"

```
2009     case 9:
2010         ritorna(p,M); //MODIFICA SPECCHIA ROBERTO
2011         break;
```

Integrazione della modifica alla funzione "switch_enigmi" per inserire la chiamata all'azione "ritorna"

```
1883     case 9:
1884         ritorna(p,M); //MODIFICA SPECCHIA ROBERTO
1885         break;
```

Metodo "resoconto" nel file Gioco.cpp

```

1558 //INIZIO MODIFICHE LEANDRA PALEMBURGI
1559     void Gioco::resoconto()
1560 {
1561     if(!percorso.pilavuota()){
1562         Pila <string> app;
1563
1564         cout << "\nRESOCONTI LUOGHI VISITATI" << endl;
1565         while ( !Luogo_p.pilavuota() )
1566         {
1567             cout << "- " << Luogo_p.leggipila() << endl;
1568             app.inpila(Luogo_p.leggipila());
1569             Luogo_p.fuoripila();
1570         }
1571         while ( !app.pilavuota() )
1572         {
1573             Luogo_p.inpila(app.leggipila());
1574             app.fuoripila();
1575         }
1576
1577         Luogo_p.fuoripila();
1578     }
1579 }
1580
1581 //FINE MODIFICHE LEANDRA PALEMBURGI

```

Modifica funzione "switch_enigmi" per l'inserimento della chiamata all'azione resoconto, in modo che funzioni sia che gli enigmi siano attivati o meno.

```

1871
1872     case 12:
1873         resoconto(); //MODIFICA PALEMBURGI LEANDRA
                     break;

```

Modifica funzione "esegui" per l'inserimento della chiamata all'azione resoconto.

```

2003
2004     case 12:
2005         resoconto(); //MODIFICA PALEMBURGI LEANDRA
                     break;

```

5.27 MODIFICA FUNZIONE AGGIORNA_TEMPO

La funzione presenta la stampa del tempo residuo in due punti diversi, cosa che causa il verificarsi di due condizioni che determinano la doppia stampa.

```

382     if(Salute.Ferito() == true) //Inizio modifica Michele Albano: se sei ferito, mostra un nuovo messaggio
383     {
384         //cout << "Tempo residuo: " << tempo << endl;
385         cout << "Tempo aggiuntivo dovuto allo Stato di Salute: " << Salute.GetPenalita() << endl;
386         cout << "Tempo residuo: " << tempo << endl;
387     }
388     else {
389         cout << "Tempo residuo: " << tempo;
390         tempo = tempo - 1 - Salute.GetPenalita(); //Fine modifica Michele Albano
391     }

```

E, proprio all'ultima riga della funzione

```
425 cout << "Tempo residuo: " << tempo << endl;
```

Il cout evidenziato nella prima immagine fa sì che qualora il personaggio non sia ferito sia stampata la sua salute. La stessa condizione, anche se non dichiarata, vale per il cout dell'ultima riga della funzione. Di seguito il risultato.

```

Sei nella tua cabina.
Uedo Raoul
Uedo:
- un letto;
- un armadietto;
- un casco;
- la tua agenda;
- un portafoglio;
- una carta di credito;
- un biglietto per lo stadio;
- 5 Euro;
- una tessera sanitaria;
- uno zaino termico;
- un documento d'identita';
- un coupon fast food blu.
Tempo residuo: 349
Tempo residuo: 347
Cosa devo fare?

```

Si modifica quindi la funzione come segue

```

388     else if(Salute.Ferito() == false) { //MODIFICA LEANDRA PALEMBURGI
389         cout << "Tempo residuo: " << tempo;
390         //tempo = tempo - 1 - Salute.GetPenalita(); //Fine modifica Michele Albano
391     }

```

Eliminando inoltre il cout alla fine della funzione, si ottiene il seguente risultato.

```

Sei nella tua cabina.
Uedo Raoul
Uedo:
- un letto;
- un armadietto;
- un casco;
- la tua agenda;
- un portafoglio;
- una carta di credito;
- un biglietto per lo stadio;
- 5 Euro;
- una tessera sanitaria;
- uno zaino termico;
- un documento d'identita';
- un coupon fast food blu.
Tempo residuo: 349
Cosa dovre fare?

```

Anche nel caso di ferita, il tempo residuo continuerà ad essere stampato una sola volta.

```

Mentre ti muovevi sei inciampato su una mattonella sporgente e ti sei slogato la
caviglia!

Provieni dal luogo: 6: Nella tua cabina

Sei nel corridoio.
Uedo Jim
Uedo:
- 10 Euro;
- 50 Euro;
Tempo aggiuntivo dovuto allo Stato di Salute: 1
Tempo residuo: 340

```

5.28 LABORATORIO ANALISI

Per implementare le modifiche di Bottiglione nel progetto base sono stati modificati i seguenti file:

- Interfaccia.cpp
- Mappa.nav
- Descrizioni (cartella)
- Astro.h
- Astro.cpp
- Oggetti.h

5.28.1 Interfaccia.cpp (123-126)

123
124	<code>string desco="\000";</code>
125	<code>desco =luogo.getDescrizione();</code>
126	<code>desco=desco.substr(0,desco.size()-1);</code> <code>cout << "\nSei " << desco << "." << endl;</code>

Le seguenti modifiche sono state apportate da Bottiglione per motivazioni non correlate all'integrazione del progetto di Scarci, ma per un problema inerente alla chiarezza del messaggio stampato a video dei luoghi.

5.28.2 Mappa.nav (1, 21, 50, 133, 134)

La prima stringa è stata modificata da 47 a 48, indicante il numero totale dei luoghi.
Il numero del luogo aggiunto è 55.

55,Nel Laboratorio Analisi, 000000190000, via 10,1,1

Il luogo "Laboratorio analisi" è raggiungibile proseguendo verso est dalla "tua cabina", salendo si arriva al pronto soccorso e poi ad est ci sarà il luogo aggiunto.

Quindi la riga contenente le direzioni del luogo "Pronto Soccorso" è stata modificata in:

19,Nel Pronto Soccorso, 000048000003, via 2,8,8

In quanto il luogo 55 corrisponde alla posizione 48.

Sono state aggiunte le seguenti righe di comando per raggiungere e abbandonare il luogo "Laboratorio analisi":

19,55, via 10, est, 1,1,1

55,19, via 10, ovest, 1,1,1

Che indicano: Partenza, Destinazione, Via, Direzione, Lunghezza, Tempo, Asfalto.

5.28.3 Descrizioni

È stato aggiunto il file 55.txt contenente la descrizione del luogo "Laboratorio analisi":

```
nel Laboratorio analisi
nel Laboratorio analisi
di nuovo nel Laboratorio analisi
nuovamente nel Laboratorio analisi
ancora nel Laboratorio analisi
```

5.28.4 Astro.h (310)

Sono state aggiunte le dichiarazioni delle procedure che eseguono le azioni.

```
310 |     void azione_157(); //analisi diabete e colesterolo
```

5.28.5 Astro.cpp

Sono state apportate le seguenti modifiche:

- Inserimento libreria (20-21)

```
20 |     #include <random> //modifiche Bottiglione Margareth
21 |     #include <chrono> // modificato da SIMONE VERISENTI
22 |
```

Libreria inclusa per poter utilizzare gli operatori all'interno della funzione

```
void Astro::azione_157()
```

- Inserimento vocaboli (492-493)

```
492 |     vocabolario.inserisci("droide", 201);  
493 |     vocabolario.inserisci("utilizza", 202);
```

- Inserimento azioni (494)

```
494 |     azioni.inserisci(4802020201, 157);
```

- Inserimento oggetti (1046)

```
1046 |     .....  
          oggetti.inserisci(Oggetto("un droide medico", 109, -48));
```

- (6570-6643)

```

6570     void Astro :: azione_157 ( )
6571     {
6572         int diabete;
6573         int colesterolo;
6574         unsigned seed = chrono::steady_clock::now().time_since_epoch().count();
6575         default_random_engine rngnumerico(seed);
6576         uniform_int_distribution <int> rangediabete(20, 400);
6577         uniform_int_distribution <int> rangecolesterolo(90, 200);
6578         int risposta_documento;
6579         bool uscita_computer= false;
6580         // Menu di scelta 0,1, ...
6581
6582         interfaccia.scrivi(" -----");
6583         interfaccia.scrivi("Sono il droide medico della nave Neutronia");
6584         interfaccia.scrivi("Quali analisi del sangue vorresti fare? ");
6585         interfaccia.scrivi("1 <- Diabete");
6586         interfaccia.scrivi("2 <- Colesterolo");
6587         interfaccia.scrivi("3 <- Saluta il droide");
6588         interfaccia.scrivi(" ----- \n");
6589
6590         cin >>risposta_documento;
6591         if(cin.good() == 0)
6592         {
6593             interfaccia.scrivi("Comando non riconosciuto... stai per uscire!");
6594             cin.clear();
6595             return;
6596         }
6597         switch (risposta_documento)
6598         {
6599             case 1:
6600                 diabete = rangediabete(rngnumerico);
6601                 interfaccia.scrivi("Diabete: ");
6602                 cout<<diabete<<endl;
6603                 if(diabete<=60){
6604                     interfaccia.scrivi("Sei in ipoglicemia");
6605                 }
6606                 else if(diabete>61&& diabete<100){
6607                     interfaccia.scrivi("Il tuo livello di glicemia è nella norma");
6608                 }
6609                 else if(diabete>100&&diabete<200){
6610                     interfaccia.scrivi("La tua glicemia è alterata o leggermente alta");
6611                 }
6612                 else if(diabete>200){
6613                     interfaccia.scrivi("Sei in iperglicemia");
6614                 }
6615                 break;
6616
6617
6618             case 2:
6619                 colesterolo = rangecolesterolo(rngnumerico);
6620                 interfaccia.scrivi("Colesterolo: ");
6621                 cout<<colesterolo<<endl;
6622
6623
6624                 if(colesterolo<130){
6625                     interfaccia.scrivi("Il livello del colesterolo è normale");
6626                 }
6627                 else if(colesterolo>130&& diabete<160){
6628                     interfaccia.scrivi("Il livello del colesterolo è alterato o leggermente alto");
6629                 }
6630
6631                 else if(colesterolo>160){
6632                     interfaccia.scrivi("Il livello del colesterolo è alto");
6633                 }
6634                 break;
6635             case 3:
6636                 uscita_computer=true;
6637                 break;
6638             default:
6639                 interfaccia.scrivi("Comando non riconosciuto... stai per uscire!");
6640                 uscita_computer=true;
6641
6642         }
6643     }

```

- (7119-7121)

```
7119    case 157:  
7120        azione_157();  
7121    break;
```

5.28.6 Dizionario.h

Le realizzazioni delle strutture dati presenti nel progetto di Bottiglione erano già presenti in maniera identica nel progetto base di Palemburgi, eccezion fatta per le realizzazioni di Coda con priorità con heap e Dizionario con array con liste di trabocco. Per il primo caso – Coda con priorità con heap – utilizzo la realizzazione presente nel progetto base. L'unica differenza con la realizzazione presente nel progetto di Bottiglione, infatti, è la dichiarazione della lunghezza massima del vettore heap. Credo sia più corretto dichiarare la lunghezza come una costante piuttosto che come una semplice variabile intera. A livello funzionale non ci sono differenze. Per il secondo caso, invece, utilizzo la realizzazione di Bottiglione perché fa uso della parola chiave *const* (che garantisce maggiore sicurezza dei dati perché non possono essere modificati) e del riferimento & per garantire un minor spreco di memoria.

Dizionario.h

```

#ifndef DIZIONARIO_H
#define DIZIONARIO_H

#include <iostream>
#include "Lista.h"
#include "Entry.h"

#include <cstdlib>

#define MAXDIM 101

using namespace std;

typedef unsigned int HashValue;

template<typename K, class E> class Dizionario
{ // E=tipo elemento, K=tipo chiave
    typedef Entry<K, E> entry;

public:
    Dizionario();
    ~Dizionario();
    void crea();
    bool appartiene(const K&); // restituisce true se l'elemento appartiene al dizionario
    void inserisci(K, E); // inserisce un elemento nel dizionario
    E recupera(const K&); // restituisce l'elemento corrispondente alla chiave (se esiste)
    void aggiorna(const K&, E&); // aggiorna l'elemento
    void cancella(const K&); // elimina l'elemento
private:
    unsigned int H(const K&); // calcola il valore hash della chiave
    unsigned int lunghezza; // lunghezza massima del dizionario
    unsigned int nelementi; // elementi presenti in un certo istante nel dizionario
    Lista<Entry<K, E> *>* table; // liste di trabocco per le entry

    HashValue hash(string);
    HashValue hash(int);

```

```
};

template<typename K, typename E>
Dizionario<K, E>::Dizionario()
{
    crea();
}

template<typename K, typename E>
Dizionario<K, E>::~Dizionario() {
    delete[] table;
}

template<typename K, typename E>
void Dizionario<K, E>::crea()
{
    lunghezza = MAXDIM;
    nelementi = 0;
    table = new Lista<entry>[MAXDIM];
}

template<typename K, typename E>
void Dizionario<K, E>::inserisci(K chiave, E element)
{
    entry e(chiave, element);
    unsigned int h = H(chiave);
    typename Lista<entry>::posizione pos = table[h].primolista();
    table[h].inslista(e, pos);
    nelementi++;
}

template<typename K, typename E>
bool Dizionario<K, E>::appartiene( const K& key)
```

```

        bool Dizionario<K, E>::appartiene( const K& key)
{
    typename Lista<entry>::posizione iter; //Lista<entry>::posizione
    bool trovato = false;
    unsigned int pos = H(key);
    iter = table[pos].primolista();
    while (!table[pos].finelista(iter) && !trovato)
    {
        if (key == table[pos].leggilista(iter).get_key())
            trovato = true;
        iter = table[pos].succlista(iter);
    }
    return trovato;
}

template<typename K, typename E>
E Dizionario<K, E>::recupera(const K& key)
{
    typename Lista<entry>::posizione iter;
    bool trovato = false;
    E elem;
    unsigned int pos = H(key);
    iter = table[pos].primolista();
    while (!table[pos].finelista(iter) && !trovato) {
        if (key == table[pos].leggilista(iter).get_key()) {
            trovato = true;
            elem = table[pos].leggilista(iter).get_value();
        }
        iter = table[pos].succlista(iter);
    }
    return elem;
}

```

```
template<typename K, typename E>
```

```

void Dizionario<K, E>::aggiorna(const K& key, E& elemento)
{
    cancella(key);
    inserisci(key, elemento);
}

template<typename K, typename E>
void Dizionario<K, E>::cancella(const K& chiave)
{
    typename Lista<entry>::posizione iter;
    bool trovato = false;
    unsigned int pos = H(chiave);
    iter = table[pos].primolista();
    while (!table[pos].finelista(iter) && !trovato) {
        if (chiave == table[pos].legglista(iter).get_key())
            trovato = true;
        else
            iter = table[pos].succlista(iter);
    }
    if (trovato)
    {
        table[pos].canclista(iter);
        nelementi--;
    }
}

template<typename K, typename E>
unsigned int Dizionario<K, E>::H(const K& chiave) {
    return (hash(chiave) % lunghezza);
}

template<typename K, typename E>
HashValue Dizionario<K, E>::hash(string str)
{
    HashValue hash = 5381;
    int l = str.length();
    for (int i = 0; i < l; i++)
        hash = hash * 33 + str[i];
    return hash;
}

template<typename K, typename E>
HashValue Dizionario<K, E>::hash(int i) {
    return i;
}

#endif // DIZ_H

```

5.29 FUNZIONE GUARDA MAPPA

Per realizzare la funzionalità sono stati modificati i seguenti file:

- Astro.h

- Astro.cpp
- Mappa.h
- Mappa.cpp

Mappa.h

```
67 //INIZIO modifiche - De Florio Cristina
68 int get_numnodi (); //restituisce il numero dei nodi presenti nella mappa
69 //FINE modifiche - De Florio Cristina
```

Riga 68 - è definito il metodo “get_numnodi” nell’header file

Mappa.cpp

```
175 //INIZIO modifiche - De Florio Cristina
176 int Mappa::get_numnodi ()
177 {
178     return numnodi;
179 }
180 //FINE modifiche - De Florio Cristina
```

Riga 176 a 179 – è aggiunto il metodo per acquisire il numero totale dei nodi presenti nel grafo (mappa) di tipo (luogo)

Astro.h

```
313 //INIZIO modifiche De Florio Cristina
314 void azione_158(Mappa &);
315 //FINE modifiche De Florio Cristina
```

Riga 314 – inserimento metodo “azione_150” nell’header file

Astro.cpp

```
627 //INIZIO modifiche De Florio Cristina
628 azioni.inserisci(100019,158);
629 //FINE modifiche De Florio Cristina
```

Riga 628 – inserimento dell’azione “guarda mappa”

```

6650 //INIZIO modifica De Florio Cristina
6651 void Astro::azione_158(Mappa &M)
6652 {
6653     int a;
6654     int j;
6655
6656     cout<<"Quale mappa vuoi visualizzare?" << endl;
6657     cout<<"1) Mappa attuale" << " 2) Mappa successiva" << endl;
6658     cin>>a;
6659     cout<< endl;
6660
6661     string dir[] = {"N","S","E","W","U","D"};
6662
6663     switch (a)
6664     {
6665         case 1:
6666             cout<< "LUOGO ATTUALE -> " << mappa.get_nome_luogo(luogo_attuale) << endl;
6667             for (j=1;j<7;j++)
6668             {
6669                 if (mappa.luogo_adiacente(luogo_attuale,j)!=0)
6670                     cout<< dir[j-1] << " : " << mappa.get_nome_luogo(mappa.luogo_adiacente(luogo_attuale,j)) << " ";
6671             }
6672             cout<< endl;
6673             break;
6674
6675         default:
6676
6677             for (int i; i < M.get_numnodi()-1;i++)
6678             {
6679                 cout<< "LUOGO -> " << mappa.get_nome_luogo(i) << endl;
6680                 for (j=1;j<7;j++)
6681
6682                     if (mappa.luogo_adiacente(i,j)!=0)
6683                         cout<< dir[j-1] << " : " << mappa.get_nome_luogo(mappa.luogo_adiacente(i,j)) << " ";
6684
6685             cout<< endl << "-----" << endl;
6686
6687             break;
6688         }
6689     }
6690 //FINE modifica De Florio Cristina

```

Riga 6650 – 6690 – inserimento della funzionalità di visualizzazione mappa

```

7170 //INIZIO modifica De Florio Cristina
7171     case 158: azione_158(M); break;
7172 //FINE modifica De Florio Cristina

```

Riga 7170 – 7172 - inserimento funzionalità nello switch/case

5.30 LUOGO CUCINA

Per la realizzazione del luogo Cucina sono state aggiunte delle azioni che hanno portato alla modifica dei file

- astro.h
- astro.cpp
- Mappa.nav

e all'aggiunta di due nuove file:

- Cucina.cpp
- Cucina.h.

MODIFICA ASTRO.H

```
//inizio modifica Costantini Andrea
void azione_159(); // mangia barretta energetica
void azione_160(); // mangia pizza
void azione_161(); // asci frigorifero
void azione_162(); // guarda televisione
void azione_163(); // preparazione cibo
void azione_164(); // mangia mela
//fine modifica Costantini Andrea
```

MODIFICA ASTRO.CPP

Modifica Vocabolario

```
497 //inizio modifica Costantini Andrea
498     vocabolario.inserisci("mela",677);
499     vocabolario.inserisci("barretta_energetica",678);
500     vocabolario.inserisci("bimby",680);
501     vocabolario.inserisci("tavolo",681);
502     vocabolario.inserisci("frigorifero",682);
503     vocabolario.inserisci("televisione",683);
504 //fine modifica Costantini Andrea
```

modifica oggetti

```
1068 //inizio modifica Costantini Andrea
1069     oggetti.inserisci(Oggetto("mela",677,49));
1070     oggetti.inserisci(Oggetto("barretta_energetica",678,49));
1071     oggetti.inserisci(Oggetto("pizza",46,49));
1072     oggetti.inserisci(Oggetto("bimby",680, -49));
1073     oggetti.inserisci(Oggetto("tavolo",681, -49));
1074     oggetti.inserisci(Oggetto("frigorifero",682,-49));
1075     oggetti.inserisci(Oggetto("televisione",683,-49));
1076 //fine modifica Costantini Andrea
```

modifica azioni

```
812 //inizio modifica Costantini Andrea
813     azioni.inserisci(4900980678,159); // mangia barretta
814     azioni.inserisci(4900980046,160); // mangia pizza
815     azioni.inserisci(4900220682,161); // asci frigorifero
816     azioni.inserisci(4900100683,162); // guarda televisione
817     azioni.inserisci(4900800680,163); // preparazione cibo
818     azioni.inserisci(4900980677,164); // mangia mela
819 //fine modifica Costantini Andrea
```

modifica switch case

```
7329 //inizio modifica Costantini Andrea
7330     case 159:
7331         azione_159();
7332         break;
7333     case 160:
7334         azione_160();
7335         break;
7336     case 161:
7337         azione_161();
7338         break;
7339     case 162:
7340         azione_162();
7341         break;
7342     case 163:
7343         azione_163();
7344         break;
7345     case 164:
7346         azione_164();
7347         break;
7348 //fine modifica Costantini Andrea
```

implementazioni delle azioni per la Cucina

```

6717 //Anzia modifica Costantini Andrea
6718 Cucina cucina = Cucina();
6719 void Astro::azione_159()
6720 {
6721     int k=oggetti.get Oggetto_index_by codice(678);
6722     if (oggetti.get Oggetto(k).get Luogo() == 0)
6723     {
6724         interfaccia.scrivi("Hai mangiato la barretta energetica");
6725         tempo=tempo-20;
6726         Salute.SetStatoSalute(Salute.GetStatoSalute()+10);
6727         interfaccia.scrivi("Hai perso 20 secondi per mangiare la barretta energetica");
6728         interfaccia.scrivi("Hai guadagnato 5 punti punti salute");
6729         oggetti.set Luogo(k,-99);
6730     }
6731     else
6732     {
6733         if (luogo_attuale==49)
6734         {
6735             interfaccia.scrivi("Per mangiare la barretta energetica, devi prima prendertela");
6736         }
6737         else
6738         {
6739             if (oggetti.get Oggetto(k).get Luogo() == -99)
6740             {
6741                 interfaccia.scrivi("Hai già mangiato la barretta, prova a mangiare qualcosa' altro");
6742             }
6743             else
6744             {
6745                 interfaccia.scrivi("Se vuoi la barretta energetica devi andare in cucina");
6746             }
6747         }
6748         fflush(stdout);
6749     }
6750 }
6751
void Astro::azione_160()
{
6752     int k=oggetti.get Oggetto_index_by codice(679);
6753     if (oggetti.get Oggetto(k).get Luogo() == 0)
6754     {
6755         interfaccia.scrivi("Hai mangiato la pizza");
6756         tempo=tempo-20;
6757         Salute.SetStatoSalute(Salute.GetStatoSalute()+5);
6758         interfaccia.scrivi("Hai perso 20 secondi per mangiare la pizza");
6759         interfaccia.scrivi("Hai guadagnato 5 punti punti salute");
6760         oggetti.set Luogo(k,-99);
6761     }
6762     else
6763     {
6764         if (luogo_attuale==49)
6765         {
6766             interfaccia.scrivi("Per mangiare la pizza, devi prima prenderla");
6767         }
6768         else
6769         {
6770             if (oggetti.get Oggetto(k).get Luogo() == -99)
6771             {
6772                 interfaccia.scrivi("Hai già divorziato la pizza, prova a mangiare qualcosa' altro");
6773             }
6774             else
6775             {
6776                 interfaccia.scrivi("Se hai voglia di pizza devi andare in cucina");
6777             }
6778         }
6779     }
6780     fflush(stdout);
6781 }
6782
6783
void Astro::azione_161()
{
6784     int k=0;
6785     k = cucina.usaFrigo();
6786     if (k!=1)
6787     {
6788         tempo=tempo-10;
6789         Salute.SetStatoSalute(Salute.GetStatoSalute()+5);
6790         interfaccia.scrivi("Hai perso 10 secondi");
6791         interfaccia.scrivi("Hai guadagnato 5 punti salute");
6792     }
6793 }
6794
6795
6796
6797
void Astro::azione_162()
{
6798     cucina.guardaTelevisione();
6799     Salute.SetStatoSalute(Salute.GetStatoSalute()-3);
6800     interfaccia.scrivi("Hai perso 3 punti salute");
6801 }
6802
6803
6804
6805
6806 void Astro::azione_163()
6807 {
6808     cucina.preparaCibo();
6809 }
6810
6811
6812
6813
6814 void Astro::azione_164()
6815 {
6816     int k=oggetti.get Oggetto_index_by codice(677);
6817     if (oggetti.get Oggetto(k).get Luogo() == 0)
6818     {
6819         interfaccia.scrivi("Hai mangiato la mela");
6820         tempo=tempo-20;
6821         Salute.SetStatoSalute(Salute.GetStatoSalute()+5);
6822         interfaccia.scrivi("Hai perso 10 secondi per mangiare la mela");
6823         interfaccia.scrivi("Hai guadagnato 5 punti punti salute");
6824         oggetti.set Luogo(k,-99);
6825     }
6826     else
6827     {
6828         if (luogo_attuale==49)
6829         {
6830             interfaccia.scrivi("Per mangiare la mela, devi prima prenderla");
6831         }
6832         else
6833         {
6834             if (oggetti.get Oggetto(k).get Luogo() == -99)
6835             {
6836                 interfaccia.scrivi("Hai già mangiato la mela, prova a mangiare qualcosa' altro");
6837             }
6838             else
6839             {
6840                 interfaccia.scrivi("Se hai voglia di mela devi andare in cucina");
6841             }
6842         }
6843     }
6844     fflush(stdout);
6845 }
6846
6847 //Fine modifica Costantini Andrea

```

MODIFICA MAPPA.NAV

8 6,Nella tua cabina,490003002122,via 3,2,2,via 11,1,1

51 56,Nella cucina,000600000000,via 11,1,1

136 6,56,Via 11,nord 1,1,1
137 56,6,_via 11,sud,1,1,1

CUCINA.H

```
#ifndef CUCINA_H_INCLUDED
#define CUCINA_H_INCLUDED
#include "Lista.h"
#include <Windows.h>
#include <sstream>
#include <string>
#include <iostream>

class Cucina
{
public:
    Cucina();
    int usaFrigo();
    void AggiungiCibo();
    void guardaTelevisione();
    void preparaCibo();

private:
    Lista<string>* CibiFrigo;
    Lista<string>* CibiPreparazione;
};

#endif // CUCINA_H_INCLUDED
```

CUCINA.CPP

Implementazione di AggiungiCibo

```
14     void Cucina::AggiungiCibo() {
15         //Cibifrigo=new Lista<string>;
16         CibiFrigo=new Lista<string>;
17         CibiPreparazione=new Lista<string>;
18
19
20         // inserimento cibi frigo
21         posizione=CibiFrigo->primolista();
22         CibiFrigo->inslista("Carote",posizione);
23         posizione=CibiFrigo->primolista();
24         CibiFrigo->inslista("Formaggio",posizione);
25         posizione=CibiFrigo->primolista();
26         CibiFrigo->inslista("Macedonia",posizione);
27         posizione=CibiFrigo->primolista();
28         CibiFrigo->inslista("Romedori",posizione);
29         //
30
31
32         // inserimento cibi bimby
33         posizione=CibiPreparazione->primolista();
34         CibiPreparazione->inslista("cipolline_in_agrodolce",posizione);
35         posizione=CibiPreparazione->primolista();
36         CibiPreparazione->inslista("insalata_russa",posizione);
37         posizione=CibiPreparazione->primolista();
38         CibiPreparazione->inslista("pure'",posizione);
39         posizione=CibiPreparazione->primolista();
40         CibiPreparazione->inslista("lasagne",posizione);
41         posizione=CibiPreparazione->primolista();
42         CibiPreparazione->inslista("marmellata",posizione);
43         posizione=CibiPreparazione->primolista();
44         CibiPreparazione->inslista("risotto",posizione);
45         posizione=CibiPreparazione->primolista();
46         CibiPreparazione->inslista("muffin",posizione);
47         posizione=CibiPreparazione->primolista();
48         CibiPreparazione->inslista("caffuccino",posizione);
49         posizione=CibiPreparazione->primolista();
50         CibiPreparazione->inslista("pesto",posizione);
51         posizione=CibiPreparazione->primolista();
52         CibiPreparazione->inslista("nutella",posizione);
53 }
```

Implementazione di usaFrigo

```
55     int Cucina::usaFrigo()
56     {
57         string i="";
58         //string xxxx="";
59         int fine=0;
60
61         if (CibiFrigo->listavuota()) {
62             cout << "Nel frigorifero non e' rimasto piu' nulla. Prova a cercare altrove." << endl;
63             return 0;
64         } else {
65
66             cout << "Hai appena aperto il frigo\n" << endl;
67             cout << "Nel frigorifero ci sono i seguenti cibi:\n" << endl;
68
69             posizione=CibiFrigo->primolista();
70
71             while (!CibiFrigo->finelista(posizione)) {
72                 cout << "- "+CibiFrigo->legglista(posizione) +";" << endl;
73                 posizione=CibiFrigo->succlista(posizione);
74             }
75             cout << "Che cibo vorresti mangiare ? \n" << endl;
76             getline(cin, i);
77
78             posizione=CibiFrigo->primolista();
79             while (!CibiFrigo->finelista(posizione) && fine==0) {
80                 if (i==CibiFrigo->legglista(posizione)) {
81                     fine=1;
82                     CibiFrigo->canclista(posizione);
83
84                 } else {
85                     posizione=CibiFrigo->succlista(posizione);
86                 }
87             }
88             if (fine==1) {
89                 cout << "Hai appena mangiato il seguente cibo : ";
90                 cout << i;
91                 cout << "\n" << endl;
92                 return 1;
93             }
94             else {
95                 cout << "Questo cibo non e' presente nel frigorifero." << endl;
96                 return 0;
97             }
98         }
99     }
100 }
```

Implementazione di guardaTelevisione

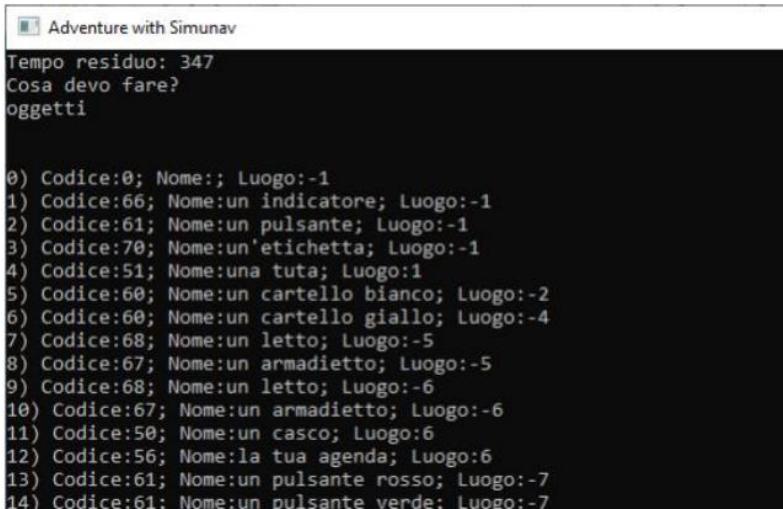
```
101     void Cucina::guardaTelevisione() {
102         cout << "Hai acceso la tv. Stai guardando il canale SPACE TV" << endl;
103         cout << "Scrivi 'Spegni' per spegnere la tv e tornare al tuo dovere" << endl;
104         string risposta="";
105         do {
106             getline(cin, risposta);
107         } while(risposta!="Spegni");
108
109         cout << "Hai spento la tv." << endl;
110
111
112     }
```

Implementazione di preparaCibo

```
114 void Cucina::preparaCibo() {
115     cout << "Quale piatto vuoi preparare ?\n" << endl;
116     posizione=CibiPreparazione->primolista();
117     int i=1;
118     string c="";
119     while (!CibiPreparazione->finelista(posizione)) {
120         cout << i;
121         cout << " - " + CibiPreparazione->legglista(posizione) + ";" << endl;
122         posizione=CibiPreparazione->succlista(posizione);
123         i++;
124     }
125
126     cout << "Risposta : " << endl;
127     getline(cin, c);
128
129     if (atoi(c.c_str()) >= 1 && atoi(c.c_str()) <= 10) {
130         int kk=1;
131         int fine=0;
132         posizione=CibiPreparazione->primolista();
133         do {
134
135             if (atoi(c.c_str())==kk) {
136                 fine =1;
137             } else {
138                 kk++;
139                 posizione=CibiPreparazione->succlista(posizione);
140             }
141         } while(!CibiPreparazione->finelista(posizione) && fine==0);
142
143         string temp=CibiPreparazione->legglista(posizione);
144         posizione=CibiFrigo->primolista();
145         CibiFrigo->inserisci(temp,posizione);
146         cout << "Hai appena preparato e messo in frigo il seguente piatto : ";
147         cout << temp << endl;
148
149     } else {
150         cout << "Piatto non disponibile";
151     }
152
153 }
154 }
```

5.31 Comando “oggetti”

E’ stato introdotto il comando “oggetti” che permette di visualizzare l’elenco di tutti gli oggetti, visualizzando per ognuno: la posizione (indice) all’interno del vettore di oggetti, il codice, il nome e il luogo in cui esso si trova. Il comando è stato introdotto tramite una apposita azione e il metodo “Astro::azione_165()”



The screenshot shows a terminal window titled "Adventure with Simunav". The user has typed "oggetti" and the system has responded with a list of 14 objects, each with a unique ID (Codice), name (Nome), and location (Luogo). The list includes various items such as a white sign, a yellow sign, a bed, a blue t-shirt, a red button, a blue button, and several other household items.

```
Tempo residuo: 347
Cosa devo fare?
oggetti

0) Codice:0; Nome:; Luogo:-1
1) Codice:66; Nome:un indicatore; Luogo:-1
2) Codice:61; Nome:un pulsante; Luogo:-1
3) Codice:70; Nome:un'etichetta; Luogo:-1
4) Codice:51; Nome:una tuta; Luogo:1
5) Codice:60; Nome:un cartello bianco; Luogo:-2
6) Codice:60; Nome:un cartello giallo; Luogo:-4
7) Codice:68; Nome:un letto; Luogo:-5
8) Codice:67; Nome:un armadietto; Luogo:-5
9) Codice:68; Nome:un letto; Luogo:-6
10) Codice:67; Nome:un armadietto; Luogo:-6
11) Codice:50; Nome:un casco; Luogo:6
12) Codice:56; Nome:la tua agenda; Luogo:6
13) Codice:61; Nome:un pulsante rosso; Luogo:-7
14) Codice:61; Nome:un pulsante verde; Luogo:-7
```

5.32 Funzionalità “zaino termico”

Nell’azione_79, l’indice rappresenta la posizione dello zaino termico all’interno dell’istanza “oggetti”. Con questa funzione è possibile visualizzare il contenuto dello zaino:

```
//Azione Zaino Termico: mostra il contenuto dello Zaino Termico
void Astro::azione_79() // ex azione 43
{
    int indice_zaino = 66;
    if(oggetti.get Oggetto(indice_zaino).get Luogo()==0)//Codice da modificare in
    base alla posizione dell'oggetto Zaino Termico
    {
        riferimento_zaino=zaino_frigo.primolista();
        if(zaino_frigo.listavuota())
            interfaccia.scrivi("Non c'e' nulla nello zaino termico. Qui puoi
                                conservare il cibo!");
        else
        {
            interfaccia.scrivi("Nello zaino termico hai messo:");
            do
            {
                if (zaino_frigo.finelist(a(riferimento_zaino))
                    cout << "\n- " <<

oggetti.get Oggetto(zaino_frigo.leggilista(riferimento_zaino)).get Nome()
                    <<
".";
                else
                    cout << "\n- " <<

oggetti.get Oggetto(zaino_frigo.leggilista(riferimento_zaino)).get Nome()
                    <<
";";
                riferimento_zaino=zaino_frigo.succlista(riferimento_zaino);
            }
            while(!zaino_frigo.finelist(a(riferimento_zaino)));
            cout<<endl;
        }
    }
    else
        interfaccia.scrivi("Non hai nessuno zaino termico! Dovresti prima
prenderlo.");
}
```

5.33 Implementazione delle capacità

```
    Implementazione di Capacita.h:  
a#ifndef CAPACITA_H_INCLUDED  
  
#define CAPACITA_H_INCLUDED  
  
  
#define MAXCAPACITA 15  
  
  
#include <iostream>  
#include <string>  
#include <stdlib.h>  
#include "Capacitasingola.h"  
#include "Lista.h"  
using namespace std;  
  
  
class Capacita {  
public:  
  
    //Operatori LISTA  
    Capacita();  
    void inserisci (string,string,int,int);           //inserisce la capacita  
    all'interno della lista  
    int get_n_capacita ();                          //restituisce il numero delle capacita all'interno della lista  
    void get_capacita (Cella<Capacita_singola*>*);      //restituisce una  
    capacita specifica  
    void set_luogo_capacita(string,int);   //setta il nuovo luogo della capacita (uniche possibilita: 0 o  
    -99)  
    void elenca_capacita ();  
    //elenca le capacita  
    string get_nome_capacita (Cella<Capacita_singola*>*); //restituisce il nome  
    int get_luogo_capacita (Cella<Capacita_singola*>*);   //restituisce il luogo  
    string get_descrizione_capacita(Cella<Capacita_singola*>*); //restituisce la descrizione  
    Lista<Capacita_singola> get_lista_capacita();        //restituisce la capacita
```

```

int get_codice_capacita(Cella<Capacita_singola>*);      //restituisce il codice
void svuota ();                                         //svuota elenco capacita

bool appartenenza (string);

private:
    Lista<Capacita_singola> capacita;
    int fine_capacita; //numero di capacita totali
};

#endif // CAPACITA_H_INCLUDED

```

Implementazione di Capacita.cpp:

```

#include "Capacita.h"

Capacita::Capacita (){
    capacita.crealista();
    fine_capacita = 1;
}

//inserimento in Lista
void Capacita::inserisci(string n,string d,int c,int l){
    if (fine_capacita<MAXCAPACITA){
        Cella<Capacita_singola>* temp = capacita.primolista();
        while(!capacita.finelista(temp)) //MODIFICA TOMAI
        {
            temp=capacita.succlista(temp);
        }
    }
}

```

```

//FINE

Capacita_singola cap(n,d,c,l);
capacita.inslista(cap, temp);
fine_capacita++;

}

}

int Capacita::get_n_capacita(){
    return fine_capacita;
}

void Capacita::get_capacita(Cella<Capacita_singola>* i){
    capacita.legglist(i).stampa_capacita_singola();
}

//MODIFICA luogo capacità presente in lista

void Capacita::set_luogo_capacita(string nome, int luogo){
    Cella<Capacita_singola>* indice = capacita.primolista();
    while(!capacita.finlista(indice)){
        if(capacita.legglist(indice).get_nome()==nome){
            Capacita_singola
            cap(get_nome_capacita(indice),get_descrizione_capacita(indice),get_codice_capacita(indice),luogo);
            capacita.canclista(indice);
            indice = capacita.primolista();
            capacita.inslista(cap, indice);
        }
        indice = capacita.succlista(indice);
    }
}

//Lettura nome capacita in LISTA

string Capacita::get_nome_capacita(Cella<Capacita_singola>* i){

```

```

        return capacita.leggilista(i).get_nome();
    }

//Lettura luogo capacità in LISTA

int Capacita::get_luogo_capacita (Cella<Capacita_singola>* i){
    return capacita.leggilista(i).get_luogo();
}

//Lettura descrizione capacità in LISTA

string Capacita::get_descrizione_capacita(Cella<Capacita_singola>* i){
    return capacita.leggilista(i).get_descrizione();
}

//Lettura codice capacità in LISTA

int Capacita::get_codice_capacita(Cella<Capacita_singola>* i){
    return capacita.leggilista(i).get_codice();
}

//Stampa elenco capacità con LISTA

void Capacita::elenca_capacita () {
    Cella<Capacita_singola>* indice = capacita.primolista();
    while (!capacita.finelista(indice)){
        if (capacita.leggilista(indice).get_luogo() == 0)
            get_capacita(indice);
        indice = capacita.succlista(indice);
    }
}

void Capacita::svuota() {
    capacita.~Lista();
    fine_capacita=0;
}

```

```
}

//Lista capacita con LISTA
Lista<Capacita_singola> Capacita::get_lista_capacita(){

    return capacita;
}

}
```

Implementazione di Capacitasingola.h:

```
#ifndef CAPACITASINGOLA_H_INCLUDED
#define CAPACITASINGOLA_H_INCLUDED

#include <iostream>
#include <string>
#include <stdlib.h>

using namespace std;

class Capacita_singola{
public: //membri pubblici della classe
    Capacita_singola(); //costruttori
    Capacita_singola(string,string,int,int);
    ~Capacita_singola(); //distruttore
    void set_luogo_capacita_singola (int); //funzione membro che setta il luogo della capacita
    string get_nome () const; //restituisce il nome della capacita
    string get_descrizione ()const; //restituisce la descrizione della capacita
    int get_luogo (); //restituisce il luogo della capacita
    int get_codice (); //restituisce il codice della capacita
    void stampa_capacita_singola(); //funzione membro che stampa la singola capacita
};

//
```

```

private: //membri privati della classe

    int codice;

    string descrizione;

    string nome;

    int luogo_capacita;

};

#endif // CAPACITASINGOLA_H_INCLUDED

```

Implementazione di Capacitasingola.cpp:

```

#include "Capacitasingola.h"

Capacita_singola::Capacita_singola() { //costruttore
    codice=0;
    luogo_capacita=0;
}

Capacita_singola::Capacita_singola(string n,string d,int c,int l){ //costruttore
    nome=n;
    descrizione=d;
    codice=c;
    luogo_capacita=l;
}

Capacita_singola::~Capacita_singola() {} //distruttore

void Capacita_singola::set_luogo_capacita_singola(int luogo) //funzione che setta il luogo
{
    luogo_capacita=luogo;
}

string Capacita_singola::get_nome ()const //restituisce il nome
{
    return nome;
}

string Capacita_singola::get_desrizione ()const //restituisce la descrizione

```

```

{
    return descrizione;
}

int Capacita_singola::get_luogo () //restituisce il luogo
{
    return luogo_capacita;
}

int Capacita_singola::get_codice() //restituisce il codice
{
    return codice;
}

void Capacita_singola::stampa_capacita_singola () //stampa a video la capacita
{
    cout<<"-" << get_nome () <<": " << get_descrizione () << endl;
}

```

Azioni aggiunte in Astro.cpp:

```

//inizio modifiche Gravina Antonio
void Astro::azione_166() //elenca capacità
{
    if (n_capacita_acquisite != 0)
    {
        interfaccia.scrivi("Possiedi come capacita' :");
        capacita.elenca_capacita();
    }
    else interfaccia.scrivi("Impossibile elencare capacita'. Il personaggio non ne conosce!");
}
//fine modifiche Gravina Antonio

//inizio modifiche Gravina Antonio
void Astro::azione_167() //elimina capacità
{
    string cap1, cap2;
    bool trovato = false;
    if (n_capacita_acquisite != 0)
    {
        interfaccia.scrivi("Possiedi come capacita' :");
        capacita.elenca_capacita();
    }
}

```

```

}

cap1 = interfaccia.leggi_stringa("Quale capacita' vuoi eliminare?\n");
Lista<Capacita_singola> cap = capacita.get_lista_capacita();
Cella<Capacita_singola> *indice = cap.primolista();
while (!cap.finelista(indice) && !trovato)
{
    cap2 = capacita.get_nome_capacita(indice);
    if (cap1 == cap2 && capacita.get_luogo_capacita(indice) == 0)
    {
        if (salito == true) interfaccia.scrivi("Non puoi dimenticare come si guida mentre stai guidando!
Scendi dal motorino prima\n");
        else
        {
            capacita.set_luogo_capacita(capacita.get_nome_capacita(indice), -99);
            n_capacita_acquisite -= 1;
            trovato = true;
            interfaccia.scrivi("Capacita' eliminata!\n");
        }
    }
    indice = cap.succlista(indice);
}
if (trovato == false) interfaccia.scrivi("Capacita' da eliminare non trovata.");
}

//fine modifiche Gravina Antonio

//inizio modifiche Gravina Antonio
void Astro::azione_168() //guarda motorino
{
    if (oggetti.get Oggetto(og).get_luogo() == 0 || oggetti.get Oggetto(og).get_luogo() == luogo_attuale)
    {
        bool trovato = false;
        string capacita_da_imparare = "guidare";
        Lista<Capacita_singola> cap = capacita.get_lista_capacita();
        Cella<Capacita_singola> *indice = cap.primolista();
        bool fine = false;
        while (!cap.finelista(indice) && fine == false)
        {
            if (capacita.get_nome_capacita(indice) == capacita_da_imparare)
            {

```

```

fine = true;
if (capacita.get_luogo_capacita(indice) == 0) trovato = true;
}
indice = cap.succlista(indice);
}
Cella<Capacita_singola> *temp;
interfaccia.scrivi("E' un normale motorino. \n");
interfaccia.scrivi("Lo guardi attentamente e noti il marchio del veicolo: Stardust! -\n");
if (!trovato)
{
if (n_capacita_acquisite < 5)
{
//acquisizione della capacita' di guidare
temp = cap.primolista();
bool fatto = false;
while (!cap.finelista(temp) && !fatto)
{
if (capacita_da_imparare == capacita.get_nome_capacita(temp))
{
capacita.set_luogo_capacita("guidare", 0);
interfaccia.scrivi("E' un modello che sapevi guidare. Navighi nella memoria e...\n");
interfaccia.scrivi("Complimenti!Hai appreso la capacita' di guidare il motorino!\n");
interfaccia.scrivi("Sperimenta cio' che hai appreso con il comando utilizza motorino!\n");
n_capacita_acquisite += 1;
fatto = true;
}
temp = cap.succlista(temp);
}
}
else interfaccia.scrivi("Non hai appreso la capacita' di guidare in quanto hai raggiunto il numero massimo di capacita' acquisibili! Riprova l'azione dopo che hai eliminato almeno una capacita'!\n");
}
}
else interfaccia.scrivi("Non capisco ");
}
//fine modifiche Gravina Antonio

//inizio modifiche Gravina Antonio
void Astro::azione_169() //leggi foglio per l'impiccato
{

```

```

//system("cls");
CLEARSCREEN(); // inserito da SIMONE VERSIENTI per compatibilità Linux/Windows
string capacita_da_controllare = "leggere";
Lista<Capacita_singola> cap = capacita.get_lista_capacita();
Cella<Capacita_singola> *indice = cap.primolista();
bool trovato = false;
//system("clear"); //linux os //Modificato da ANTONIO PASTORELLI
string risp;

while (!cap.finelista(indice) && trovato == false)
{
    if (capacita_da_controllare == capacita.get_nome_capacita(indice))
    {
        trovato = true;
        if (capacita.get_luogo_capacita(indice) == 0)
        {
            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_inovinata;
                    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!!!");

}

else interfaccia.scrivi("C'e' qualcosa scritto... potrebbe essere interessante ma ti manca la capacita'
di leggere! Riprova quando avrai ottenuto tale capacita'!");

}

indice = cap.succlista(indice);

}

}

//fine modifiche Gravina Antonio

//inizio modifiche Gravina Antonio

void Astro::azione_170() //guarda poster
{
    string risposta;
    string capacita_da_controllare;
    Lista<Capacita_singola> cap = capacita.get_lista_capacita();
    Cella<Capacita_singola> *indice = cap.primolista();
    int trovato = 0;
    while (!cap.finelista(indice) && trovato == 0)           //Se leggere non è nelle mie capacità, non posso
leggere ciò che è scritto sul poster
    {
        if ("leggere" == capacita.get_nome_capacita(indice))
        {
            if (capacita.get_luogo_capacita(indice) == 0)
            {
                trovato = 1;
                indice = cap.primolista();
            }
            else trovato = 2;
        }
        else indice = cap.succlista(indice);
    }
    if (trovato == 1)
    {
        interfaccia.scrivi("E' un poster attaccato alla parete; c'e' una nota scritta da me: \n");
        interfaccia.scrivi("- esistono alcune azioni chiamate capacita' che possono essere apprese solo in
alcuni luoghi -");
    }
}

```

```

interfaccia.scrivi("- puoi visualizzare le capacita' in tuo possesso o cancellarle in qualsiasi momento -");
interfaccia.scrivi("- per esempio, ecco alcune capacita' disponibili: -\n");
interfaccia.scrivi("1) Piegarsi");
interfaccia.scrivi("2) Premere");

do
{
    risposta = interfaccia.leggi_stringa("Quale capacita' ti piacerebbe apprendere fra queste?\n");
    trovato = 1;
    if (risposta == "1" || risposta == "piegarsi") capacita_da_controllare = "piegarsi";
    else if (risposta == "2" || risposta == "premere") capacita_da_controllare = "premere";
    else trovato = 0;
}
while (trovato == 0);

int esiste = 0;
while (!cap.finlista(indice) && esiste == 0)
{
    if (capacita_da_controllare == capacita.get_nome_capacita(indice))
    {
        if (capacita.get_luogo_capacita(indice) == 0) esiste = 1;
        else esiste = 2;
    }
    else indice = cap.succlista(indice);
}

if (esiste == 1) interfaccia.scrivi("Hai gia' appreso questa capacita'!!!\n");
else
{
    if (n_capacita_acquisite < 5)
    {
        if (capacita_da_controllare == capacita.get_nome_capacita(indice))
        {
            capacita.set_luogo_capacita(capacita.get_nome_capacita(indice), 0);
            interfaccia.scrivi("Complimenti! Hai appreso la capacita' di " + capacita_da_controllare + "!\n");
            n_capacita_acquisite += 1;
        }
    }
}

```

```

        }

        else interfaccia.scrivi("Non hai appreso la capacita' di " + capacita_da_controllare + " perche' hai
raggiunto il numero massimo di capacita' apprese.\n");
    }

}

else interfaccia.scrivi("Leggere e' una delle capacita' piu' importanti del gioco, e hai deciso di eliminarla.
E adesso come te la caverai?");

}

//fine modifiche Gravina

//inizio modifiche Gravina Antonio
void Astro::azione_171() //parla al vagabondo
{
    string risp, cap1, cap2;

    interfaccia.scrivi("Ciao, sono molto affamato. Ho voglia di una mela..."); 
    do risp = interfaccia.leggi_stringa("Me la puoi dare? [S/N]"); 
    while (risp != "SI" && risp != "NO" && risp != "si" && risp != "no" && risp != "n" && risp != "s");
    if ((risp == "si" || risp == "s" || risp == "Si") && (n_capacita_acquisite < 5))
    {
        int k = oggetti.get_oggetto_indice_by_codice(677);
        if(oggetti.get_luogo(k) == 0)
        {
            oggetti.set_luogo(k, -99);
            interfaccia.scrivi("Visto che mi hai aiutato ti ricompensero'. Scegli fra una di queste capacita'
base:");
            cap1 = interfaccia.leggi_stringa("-aprire\n-digitare\n-leggere\n-scrivere");
            Lista<Capacita_singola> cap = capacita.get_lista_capacita();
            Cella<Capacita_singola> *indice = cap.primolista();
            bool trovato = false;
            while (!cap.finelista(indice) && !trovato)
            {
                cap2 = capacita.get_nome_capacita(indice);
                if (cap1 == cap2)
                {
                    trovato = true;
                    if (capacita.get_luogo_capacita(indice) == 0) interfaccia.scrivi("Hai gia' questa capacita' base.
Se vuoi parlarmi di nuovo porta altre mele!");
                }
            }
        }
    }
}

```

```

        n_capacita_acquisite++;
        capacita.set_luogo_capacita(cap1, 0);
        interfaccia.scrivi("Fatto. Se hai bisogno di me portami altre mele!");
    }
}
indice = cap.succlista(indice);
}
}
else interfaccia.scrivi("Non hai mele con te...");
}
else
{
    if ((risp == "si" || risp == "s" || risp == "Si") && (n_capacita_acquisite == 5)) interfaccia.scrivi("Ora che ci penso, e' meglio se prima cancelli una capacita'...");
}
}

//fine modifiche Gravina Antonio

```

Modifiche in Astro.h:

```

//inizio modifiche Gravina Antonio
void azione_166(); // elenca capacità
void azione_167(); // elimina capacità
void azione_168(); // guarda motorino
void azione_169(); // leggi foglio
void azione_170(); // guarda poster
void azione_171(); //parla vagabondo
//fine modifiche Gravina Antonio

```

Modifiche in Gioco.h:

```

#include "Capacita.h"

//inizio modifiche Gravina Antonio
int n_capacita_acquisite;
Capacita capacita;
//fine modifiche Gravina Antonio

```

Modifiche in Gioco.cpp:

```

//inizio modifiche Gravina Antonio
void Gioco::usa_motorino()
{
    string capacita_da_controllare = "guidare";

```

```

Lista<Capacita_singola> cap = capacita.get_lista_capacita();
Cella<Capacita_singola> *indice = cap.primolista();
while (!cap.finelista(indice))
{
    if (capacita_da_controllare == capacita.get_nome_capacita(indice))
    {
        if (oggetti.get_oggetto(og).get_luogo() != 0)
        {
            interfaccia.scrivi("Non hai il motorino.");
            stringa_risposta = "non avevi il motorino.";           //Modifica PMF(storia)
            storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
        }
        else
        {
            if (capacita.get_luogo_capacita(indice) == 0)
            {
                if (motorino.get_livello_benzina() == 0)
                {
                    interfaccia.scrivi("Non hai carburante, vai a fare benzina");
                    stringa_risposta = "non avevi carburante.";           //Modifica PMF(storia)
                    storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
                }
                else
                {
                    interfaccia.scrivi("Sei sul motorino! ora ti sposti più velocemente.");
                    salito = true;

                    stringa_risposta = "eri sul motorino.";
                    storia_gioco.insStoria(stringa_comando, stringa_risposta);
                }
            }
            else
                interfaccia.scrivi("Forse ti manca una capacita'. Prova ad osservare il motorino...\n");
        }
    }
    indice = cap.succlista(indice);
}
//fine modifiche Gravina Antonio

```

```

//Modifica PMF(agenda)
//Inserimento azioni relative ad agenda

//cancella agenda
void Gioco::azione_46()
{
    long size;
    Agenda agenda;
    if(oggetti.get Oggetto(12).get_luogo() != 0 )
        interfaccia.scrivi("Non ce l'hai.");
    else
    {
        ifstream check ("codici.txt");
        if (!check)
            cout <<"L'agenda e' vuota. Devi prima scriverci dentro. Prova il comando 'scrivi agenda'\n\n";
        else
            agenda.Cancella();
    }

    stringa_risposta = "hai cancellato l'agenda.";           //Modifica PMF(storia)
    storia_gioco.insStoria(stringa_comando, stringa_risposta); //Modifica PMF(storia)
}

//inizio modifiche Gravina Antonio
//scrivi agenda
void Gioco::azione_47()
{
    Lista<Capacita_singola> cap = capacita.get_lista_capacita();
    Cella<Capacita_singola> *indice = cap.primolista();
    string capacita_da_controllare = "scrivere";
    Agenda agenda;
    if (oggetti.get Oggetto(12).get_luogo() != 0 )
        interfaccia.scrivi("Non ce l'hai.");
    else
    {
        while (!cap.finlista(indice))
        {
            if (capacita_da_controllare == capacita.get nome_capacita(indice))
            {

```

```

        if (capacita.get_luogo_capacita(indice) == 0)
        {
            agenda.Scrivi();
            stringa_risposta = "hai scritto l'agenda.";
            storia_gioco.insStoria(stringa_comando, stringa_risposta);
        }
        else
            interfaccia.scrivi("Non puoi scrivere! Perche' hai voluto scordare questa capacita'? E adesso?\n");
    }
    indice = cap.succlista(indice);
}
}

//fine modifiche Gravina Antonio

//inizio modifiche Gravina Antonio
//leggi agenda
void Gioco::azione_48()
{
    Lista<Capacita_singola> cap = capacita.get_lista_capacita();
    Cella<Capacita_singola> *indice = cap.primolista();
    string capacita_da_controllare = "leggere";
    Agenda agenda;
    if(oggetti.get Oggetto(12).get_luogo() != 0)
        interfaccia.scrivi("Non ce l'hai.");
    else
    {
        while(!cap.finelista(indice))
        {
            if (capacita_da_controllare == capacita.get_nome_capacita(indice))
            {
                if (capacita.get_luogo_capacita(indice) == 0)
                {
                    ifstream check ("codici.txt");
                    if (!check)
                        cout << "L'agenda e' vuota. Devi prima scriverci dentro. Prova il comando 'scrivi agenda'\n\n";
                    else
                        agenda.Leggi();
                }
            }
        }
    }
}

```

```
else
    interfaccia.scrivi("Non puoi leggere! Perche' hai voluto scordare questa capacita'? E
adesso?\n");
}
indice = cap.succlista(indice);
}
}
//Modifica PMF: fin qui.
```

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.legglista(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>::scrivlista(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 preclistा (posizione)const;
    void inslista (tipoelem,posizione &);
    void canclistा (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>::leggilista(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>::canclista(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 finalista(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.finalista(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    }

```

```

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         succlist(a(p)->setPrec(prelista(p));
164     }
165 }
166
167 p = p->getSucc();
168 delete temp;
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 getElem() const;
15     void setElem(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>::getElem() 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 finelista(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

```

6.3.2 Pila con cursori

6.3.2.1 Pila.h

```

1. /*
2.
3. Data: Anno Accademico 2018/19
4. */
5.
6. #ifndef _PILA_CON_CURSORI_H
7. #define _PILA_CON_CURSORI_H
8.
9. #include <iostream>
10.#include<assert.h>
11.#include <string>
12.
13.template <class tipo
14.class Pila{
15.// Parte pubblica
16.public:
17.    typedef tipo tipoelem;
18.    Pila();                                // Costruttore
19.    ~Pila();                               // Distruttore
20.
21.// Operatori

```

```

22. void creapila();
23. bool pilavuota() const;
24. void inpila(tipoelem dato);
25. void fuoripila();
26. tipo leggipila() const;
27.
28. // Parte privata
29. private:
30.     static const int lung_max = 1024;      // lunghezza massima della pila
31.     typedef int rif;
32.     struct Cella{
33.         tipoelem dato;
34.         rif successivo;
35.     };
36.     Cella spazio[lung_max];
37.     rif testa;
38. };
39.
40. // Costruttore
41. template <class tipo>
42. Pila<tipo>::Pila(){
43.     creapila();
44. }
45.
46. // Distruttore
47. template <class tipo>
48. Pila<tipo>::~Pila(){
49. }
50.
51. template <class tipo>
52. void Pila<tipo>::creapila(){
53.     for(int i=lung_max; i>=0; i--){
54.         spazio[i].successivo = i-1;
55.     }
56.     testa = -1;
57. }
58.
59. // verifica se la pila è vuota
60. template <class tipo>
61. bool Pila<tipo>::pilavuota() const{
62.     return(testa == -1);
63. }
64.
65. // Aggiunge un elemento alla pila
66. template <class tipo>
67. void Pila<tipo>::inpila(tipoelem dato){
68.     testa++;
69.     assert(testa<lung_max);
70.     spazio[testa].dato = dato;
71. }
72.
73. // Rimuove un elemento dalla pila
74. template <class tipo>
75. void Pila<tipo>::fuoripila(){
76.     assert(!pilavuota());
77.     testa = spazio[testa].successivo;
78. }
79.

```

```

80. // Legge l'elemento in testa della pila
81. template <class tipo>
82. tipo Pila<tipo>::leggipila() const{
83.     assert(!pilavuota());
84.     return spazio[testa].dato;
85. }
86.
87. #endif // _PILA_CON_CURSORI_H

```

6.3.3 Pila con lista

6.3.3.1 Pila.h

```

/*
* Pila.h
*
* Created on: 10/ago/2015
* Author: GiovanniCastellana
*/

// strutturadatiGiovanniCastellana
#ifndef PILA_H_
#define PILA_H_

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

usingnamespace std;

//realizzazionediunapiladinamicautilizzandounalistadinamicamondenariezionale

template<class P> class Pila
{
public:
//definizioniditipo
typedef P tipoelem;
typedef Cella<P>* posizione;

//costruttori
Pila();
Pila(const Pila&);

//distruttoredi default

//operatoridispecifica
void creapila();
bool pilavuota() const;
tipoelem leggipila() const;
void inpila(tipoelem);

```

```

voidfuoripila();
voidsvuota(); //operatore ausiliare

//sovraffacciai
voidoperator =(const Pila& );
booloperator ==(Pila& );

private :
Lista<P>pila; //lapila è difattounalista

friendostream&operator<<(ostream& os, Pila<P>& p) //sovraffacciai
output
{
os<<p.pila;
return(os);
}

};

template<class P>Pila<P>::Pila() //costruttore generico
{
creapila();
}

template<class P>Pila<P>::Pila(const Pila<P>& p)
//costruttore di copia
{
creapila();
pila=p.pila;
}

template<class P>void Pila<P>::creapila() //crea la pila
{
pila.crealista();
}

template<class P>bool Pila<P>::pilavuota() const //restituisce true
se la pila è vuota, false altrimenti
{
return (pila.listavuota());
}

template<class P>P Pila<P>::leggipila() const //legge l'elemento in
testa alla pila
{
if (!pilavuota()) //precondizione pila non vuota
return (pila.legglist(a));
}

template<class P>void Pila<P>::inpila(tipoelem a)
//in pila un elemento

```

```

{
  posizione i=pila.primolista();
  pila.inslista(i,a);
}

template<class P>void Pila<P>::fuoripila() //estrage l'elemento in
testa
{
  posizione i=pila.primolista();
  if (!pilavuota()) //precondizionepila non vuota
  pila.canclista(i);
}

//sovrcarichi

template<class P>void Pila<P>::operator =(const Pila<P>& p)
//assegnamento
{
  pila=p.pila;
}

template<class P>bool Pila<P>::operator ==(Pila<P>& p) //uguaglianza
{
  return (pila==p.pila);
}

//operatoreausiliare

template<class P>void Pila<P>::svuota() //svuotalapila
{
  while (!pilavuota())
  {
    fuoripila();
  }
}

#endif/* PILA_H_ */

```

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

```

1. /*
2.      Realizzazione: Coda con puntatori
3.      Autore: Davide De Salvo
4. */
5.
6.
7. #ifndef CODA_H_INCLUDED
8. #define CODA_H_INCLUDED

```

```

9. #include "Cella.h"
10.
11.
12.
13. template <class tipoelem>
14. class Coda
15. {
16. public:
17.
18.     Coda();
19.     ~Coda();
20.     void creacoda();
21.     bool codavuota() const;
22.     tipoelem leggicoda() const;
23.     void fuoricoda();
24.     void incoda(tipoelem);
25. private:
26.     typedef Cella<tipoelem>* posizione;
27.     posizione testa;
28.     posizione fondo;
29. };
30.
31.
32.
33.
34. template <class tipoelem>
35. Coda<tipoelem>::Coda()
36. {
37.     creacoda();
38. }
39.
40. template <class tipoelem>
41. Coda<tipoelem>::~Coda()
42. {
43.     while(!codavuota())
44.     {
45.         fuoricoda();
46.     }
47. }
48.
49. template <class tipoelem>
50. void Coda<tipoelem>::creacoda()
51. {
52.     testa=nullptr;
53.     fondo=nullptr;
54. }
55.
56. template <class tipoelem>
57. bool Coda<tipoelem>::codavuota() const
58. {
59.     return (testa==nullptr);
60. }
61.
62. template <class tipoelem>
63. tipoelem Coda<tipoelem>::leggicoda() const
64. {
65.     return testa->leggicella();
66. }
67.
68. template <class tipoelem>
69. void Coda<tipoelem>::fuoricoda()
70. {
71.     posizione temp=testa;
72.     testa=testa->leggisucc();
73.     delete temp;
74. }
75.
```

```

76. template <class tipoelem>
77. void Coda<tipoelem>::incoda(tipoelem e)
78. {
79. posizione temp=new Cella<tipoelem>;
80. temp->scrivicella(e);
81. temp->scrivisucc(nullptr);
82. if(!codavuota())
83. {
84. fondo->scrivisucc(temp);
85. fondo=temp;
86. }
87. else
88. {
89. testa=temp;
90. fondo=temp;
91. }
92. }
93.
94.
95.
96. #endif // CODA_H_INCLUDED

```

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) //sovraffatto  
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

```
/**  
 * @file Prioricoda.h  
 * Definizione della struttura dati "Coda con priorità"  
 * @note Realizzazione tramite lista ordinata  
 * @author Gravina Antonio  
 * @date Anno Accademico 2021/2022  
 */  
  
#ifndef _PRIORICODA_H  
#define _PRIORICODA_H  
  
#include "ListaOrdinata.h"  
#include "Priorielem.h"  
#include <iostream>  
#include <cstdlib>  
  
using namespace std;  
  
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 insiemistici
bool insiemevuoto() const;
bool appartiene(tipoelem) const;

private:
    ListaOrdinata<tipoelem> prioricoda;
};

template<class P> Prioricoda<P>::Prioricoda()
{
    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
{
    if(!appartiene(a))
        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
        m = prioricoda.leggilista(prioricoda.primolista());
    return m;
}

template<class P> void Prioricoda<P>::cancellamin() //elimina il minimo dalla coda
{
    if (!prioricoda.listavuota()) //precondizione coda non vuota
    {
        Cella<tipoelem>* 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())

    {
        Cella<tipoelem>* indice = prioricoda.primolista();

        while (!prioricoda.finelista(indice) && !trovato)

        {
            if (prioricoda.leggilista(indice) == a)

                trovato = true;

            else indice = prioricoda.succlista(indice);

        }

    }

    return(trovato);
}

#endif

```

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     Albero();
23     //COSTRUTTORE DI COPIA
24     Albero(const Albero&);
25     //DISTRUTTORE
26     virtual ~Albero();
27     //METODO CHE CREA ALBERONARIO
28     void creaalbero();
29     //METODO CHE RESTITUISCE VERO SE L'ALBERO È VUOTO, FALSO ALTRIMENTI
30     bool alberovuoto() const;
31     //METODO CHE INSERISCE UN NODO NULLO COME RADICE
32     void insradice();
33     //METODO CHE RESTITUISCE IL NODO DELLA RADICE
34     nodo radice() const;
35     //METODO CHE RESTITUISCE IL NODO PADRE DEL NODO PASSATO
36     nodo padre(nodo);
37     //METODO CHE RESTITUISCE VERO SE IL NODO PASSATO È FOGLIA (QUINDI NON HA FIGLI), FA
38     LSO ALTRIMENTI
39     bool foglia(nodo) const;
40     //METODO CHE RESTITUISCE IL NODO PRIMOFIGLIO DI UN NODO PASSATO
41     nodo primofiglio(nodo) const;
42     //METODO CHE RESTITUISCE VERO SE IL NODO PASSATO NON HA FRATELLI SUCCESSIVI, FALSO
43     ALTRIMENTI
44     bool ultimofratello(nodo) const;
45     //METODO CHE RESTITUISCE IL NODO FRATELLO SUCCESSIVO DEL NODO PASSATO
46     nodo succfratello(nodo)const;
47     //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO C
48     OME NODO PRIMOFIGLIO DEL NODO PASSATO
49     void insprimosottoalbero(nodo, Albero&);
50     //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO C
51     OME UN NODO FRATELLO SUCCESSIVO DEL NODO PASSATO
52     void inssottoalbero(nodo, Albero&);
53     //METODO CHE PASSATO UN NODO CANCELLA LO STESSO E TUTTI I SUOI DISCENDENTI
54     void cancottoalbero(nodo);
55     //METODO CHE SCRIVE L'ETICHETTA DI TIPO TIPOELEM ALL'INTERNO DEL NODO PASSATO
56     void scrivinodo(nodo&, tipoelem);
57     //METODO CHE RESTITUISCE L'ETICHETTA DEL NODO PASSATO
58     tipoelem legginodo(nodo) const;
59
60 private:
61
62     BinAlbero<tipoelem> tree;
63     //METODO CHE COPIA UN ALBERO PASSATO PARTENDO DAL NODO SINISTRO
64     void copia_albero_sx(const BinAlbero<tipoelem>&, const nodo&, nodo);
65     //METODO CHE COPIA L'ALBERO PASSATO PARTENDO DAL NODO DESTRO
66     void copia_albero_dx(const BinAlbero<tipoelem>&, const nodo&, nodo);
67 };
68
69
70
71 template <class tipoelem>
72     Albero<tipoelem>::Albero(){
73     creaalbero();
74 }
75
76 template <class tipoelem>
77     Albero<tipoelem>::Albero(const Albero& a){
78     if(!a.tree.binalberovuoto())
79     {
80         tree.insbinradice();
81         nodo n = tree.binradice();
82         nodo secondo_albero = a.tree.binradice();
83         tree.scrivinodo(n, a.tree.legginodo(secondo_albero));
84
85         if(a.tree.sinistruvoto(a.tree.binradice()))
86         {
87             if(!a.tree.destrovuto(a.tree.binradice()))
88                 copia_albero_dx(a.tree,a.tree.figliodestro(a.tree.binradice()),tree
89 .binradice());
90     }
91 }
```

```

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     //RICHIAMA AUTOMATICAMENTE IL DISTRUTTORE
94     //DELL'ALBERO BINARIO
95 }
96
97 template <class tipoelem>
98 void Albero<tipoelem>::creaalbero(){
99     //RICHIAMA 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.sinistrovuoto(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{
135     return tree.figliosinistro(n);
136 }
137
138 template <class tipoelem>
139 bool Albero<tipoelem>::ultimofratello(typename Albero<tipoelem>::nodo n) const{
140     if(tree.destrovuoto(n))
141         return true;
142     else
143         return false;
144 }
145
146 template <class tipoelem>
147 typename Albero<tipoelem>::nodo Albero<tipoelem>::succfratello(typename Albero<tipoelem>
148 ::nodo n) const{
149     return tree.figliodestro(n);

```

```

147    }
148
149    template<class tipoelem>
150    void Albero<tipoelem>::copia_albero_sx(const BinAlbero<tipoelem>& other, const nodo& ra
dice_sottoalbero, nodo nodo_padre)
151    {
152        tree.insfigliosinistro(nodo_padre);
153        nodo sx = tree.figliosinistro(nodo_padre);
154        tree.scrivinodo(sx, other.legginodo(radice_sottoalbero));
155
156        if(!other.sinistruvoto(radice_sottoalbero))
157            copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), sx);
158        if(!other.destrovuto(radice_sottoalbero))
159            copia_albero_dx(other, other.figliodestro(radice_sottoalbero), sx);
160    }
161
162    template <class tipoelem>
163    void Albero<tipoelem>::copia_albero_dx(const BinAlbero<tipoelem>& other, const nodo& ra
dice_sottoalbero, nodo nodo_padre)
164    {
165        tree.insfigliodestro(nodo_padre);
166        nodo dx = tree.figliodestro(nodo_padre);
167        tree.scrivinodo(dx, other.legginodo(radice_sottoalbero));
168
169        if(!other.sinistruvoto(radice_sottoalbero))
170            copia_albero_sx(other, other.figliosinistro(radice_sottoalbero), dx);
171        if(!other.destrovuto(radice_sottoalbero))
172            copia_albero_dx(other, other.figliodestro(radice_sottoalbero), dx);
173    }
174
175    template <class tipoelem>
176    void Albero<tipoelem>::insprimosottoalbero(typename Albero<tipoelem>::nodo n, Albero& a
){
177        nodo temp;
178        if(tree.sinistruvoto(n)){
179            copia_albero_sx(a.tree,a.tree.binradice(),n);
180        }
181        else{
182            Albero<tipoelem> T;
183            T.insradice();
184            temp=T.radice();
185            T.scrivinodo(temp,tree.legginodo(tree.figliosinistro(n)));
186
187            if(tree.sinistruvoto(tree.figliosinistro(n))){
188                if(!tree.destrovuto(tree.figliosinistro(n))){
189                    copia_albero_dx(tree,tree.figliodestro(tree.figliosinistro(n)),temp
);
190                }
191            }
192            else{
193                copia_albero_sx(tree,tree.figliosinistro(tree.figliosinistro(n)),temp);
194            }
195
196            tree.cancsottobinalbero(tree.figliosinistro(n));
197
198            copia_albero_sx(a.tree,a.tree.binradice(),n);
199            temp=tree.figliosinistro(n);
200            copia_albero_dx(T.tree,T.radice(),temp);
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
ratello successivo
208        di n. n non è la radice*/

```

```

209     nodo temp;
210     if(n!=tree.binradice()){
211         if(tree.destrovuoto(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.destrovuoto(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     }
240
241     template <class tipoelem>
242     void Albero<tipoelem>::cancottoalbero(typename Albero<tipoelem>::nodo n){
243         /* L'albero è ottenuto togliendo il sottoalbero di radice n e tutti i suoi discende
244 ntii*/
245         tree.cancottobinalbero(n);
246     }
247
248     template <class tipoelem>
249     void Albero<tipoelem>::scrivinodo(typename Albero<tipoelem>::nodo& n, tipoelem elem){
250         tree.scrivinodo(n,elem);
251     }
252
253     template <class tipoelem>
254     tipoelem Albero<tipoelem>::legginodo(typename Albero<tipoelem>::nodo n) const{
255         return tree.legginodo(n);
256     }
257
258 #endif // ALBERONARIO_H

```

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 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 TIPOELEM
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.
61.     private:
62.         Nodo radice;
63.     };
64.
65.     template <class TIPOETICHETTA>
66.     BinAlbero<TIPOETICHETTA>::BinAlbero(){
67.         creabinalbero();
68.         /* if(radice!=nil)
69.             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. template <class TIPOETICHETTA>
92.     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binpadre(Nodo pad
93.     re)const{
94.         return padre->getPadre();
95.     }
96. template <class TIPOETICHETTA>
97.     bool BinAlbero<TIPOETICHETTA>::sinistrovuoto(Nodo sx)const{
98.         if(sx->getFiglioSx()==nil)
99.             return true;
100.        else
101.            return false;
102.    }
103.
104.    template <class TIPOETICHETTA>
105.        bool BinAlbero<TIPOETICHETTA>::destruovoto(Nodo dx)const{
106.            if(dx->getFiglioDx()==nil)
107.                return true;
108.            else
109.                return false;
110.        }
111.
112.    template <class TIPOETICHETTA>
113.        typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliosini
114.        stro(Nodo sx)const{
115.            return sx->getFiglioSx();
116.        }
117.    template <class TIPOETICHETTA>
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.
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(!sinistruvuoto(r))
147.                 cancsottobinalbero(r->getFiglioSx());
148.
149.             if(!destruvuoto(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.
169.     template <class TIPOETICHETTA>
170.         TIPOETICHETTA BinAlbero<TIPOETICHETTA>::legginode(Nodo n)const{
171.             return n->getEtichetta();
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.
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<TIPOETICHETTA>::Cella_Binalbero();
37         //OVERLOAD DELL'OPERATORE =
38         void operator=(const Cella_Binalbero<TIPOETICHETTA> );
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    template <class TIPOETICHETTA>
109    Cella_Binalbero<TIPOETICHETTA>::Cella_Binalbero(const Cella_Binalbero& c){
110        etichetta=c.getEtichetta();
111        FiglioDx=c.getFiglioDx();
112        FiglioSx=c.getFiglioSx();
113        Padre=c.getPadre();
114    }
115
116    template <class TIPOETICHETTA>
117    void Cella_Binalbero<TIPOETICHETTA>::operator=(const Cella_Binalbero& c ){
118        etichetta=c.getEtichetta();
119        Padre=c.getPadre();

```

```
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
36 private:
37     nodo root; //nodo radice
38 };
39
40 template <class T> Albero<T>::Albero()
41 {
42     creaalbero();
43 }
44
45 template <class T> Albero<T>::~Albero() {}
46
47 template <class T> void Albero<T>::creaalbero()
48 {
49     root=NULL;
50 }
51
52 template <class T> bool Albero<T>::alberovuoto() const
53 {
54     return(this->root==NULL);
55 }
56
57 template <class T> void Albero<T>::insradice()
58 {
59     if(alberovuoto())
60     {
```

```

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
139     delete n;
140 }
141
142 template <class T>
143 void Albero<T>::scrivinodo(nodo n, tipoelem a){
144     n->setetichetta(a);
145 }
146
147 template <class T>
148 T Albero<T>::legginodo(nodo n){
149     return (n->getetichetta());
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(Cella*);
17     Cella* getprimofiglio();
18     void setfratellosucc(Cella*);
19     Cella* getfratellosucc();
20     void setpadre(Cella*);
21     Cella* 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

```

6.6.3 ALBERO N-ARIO REALIZZATO CON VETTORE DI PADRI

```

#ifndef N_ARIOVETTOREPADRI_H_
#define N_ARIOVETTOREPADRI_H_

```

```

#include <iostream>
using namespace std;

```

```

template<class tipoelem>
class alberonv {

public:
typedef int nodo;

alberonv();
void creaalbero();
bool alberovuoto();
bool foglia(nodo);
bool ultimofratello(nodo);
nodo radice();
nodo padre(nodo);
nodo primofiglio(nodo);

nodo succfratello(nodo);
tipoelem legginodo(nodo);
void scrivinodo(nodo, tipoelem);
void insradice(nodo);
void insprimosottoalbero(alberonv, nodo);

void inssottoalbero(alberonv, nodo);
void cancsottoalbero(nodo);

private:
typedef struct Node {
int padre;
tipoelem etichetta;
} node;

node *tree;
int dimtree;
int nodialbero;
};

template<class tipoelem>
alberonv<tipoelem>::alberonv() {
    creaalbero();
}

template<class tipoelem>
void alberonv<tipoelem>::creaalbero() {
    tree = new struct Node[dimtree];
    for(int i = 0; i < dimtree; i++)
        tree[i].padre = -1;

    nodialbero = 0;
}

template<class tipoelem>
bool alberonv<tipoelem>::alberovuoto() {
    return(nodialbero == 0);
}

template<class tipoelem>
bool alberonv<tipoelem>::foglia(typename alberonv<tipoelem>::nodo nod) {
    bool esito = true;
    for(int i = nod + 1; (i < dimtree) && esito; i++) {

```

```

        if(tree[i].padre == nod)
            esito = false;
    }
    return esito;
}

template<class tipoelem>
bool alberonv<tipoelem>::ultimofratello(typename alberonv<tipoelem>::nodo nod) {
    bool esito = false;
    for(int i = nod + 1; (i < nodialbero) && !esito; i++) {
        if(tree[i].padre == tree[nod].padre)
            esito = true;
    }
    return esito;
}

template<class tipoelem>
typename alberonv<tipoelem>::nodo alberonv<tipoelem>::radice() {

    return 0;
}

template<class tipoelem>
typename alberonv<tipoelem>::nodo alberonv<tipoelem>::padre(typename alberonv<tipoelem>::nodo nod) {
    return(tree[nod].padre);
}

template<class tipoelem>
typename alberonv<tipoelem>::nodo alberonv<tipoelem>::primofiglio(typename alberonv<tipoelem>::nodo nod) {
    bool trovato = false;
    nodo figlio;
    for(int i = nod; (i < nodialbero) && !trovato; i++) {
        if(tree[i].padre == nod) {
            figlio = i;
            trovato = true;
        }
    }
    return figlio;
}

template<class tipoelem>
typename alberonv<tipoelem>::nodo alberonv<tipoelem>::succfratello(typename alberonv<tipoelem>::nodo nod) {
    bool trovato = false;
    nodo fratello;
    for(int i = nod + 1; (i < nodialbero) && !trovato; i++) {

        if(padre(i) == padre(nod)) {
            trovato = true;
            fratello = i;
        }
    }
    return fratello;
}

template<class tipoelem>
tipoelem alberonv<tipoelem>::legginodo(typename alberonv<tipoelem>::nodo nod) {
    return(tree[nod].etichetta);
}

```

```

}

template<class tipoelem>
void alberonv<tipoelem>::scrivinodo(typename alberonv<tipoelem>::nodo nod, tipoelem elem) {
    tree[nod].etichetta = elem;
}

template<class tipoelem>
void alberonv<tipoelem>::insradice(typename alberonv<tipoelem>::nodo nod) {
    tree[0].padre = 0;
    nodialbero++;
}

template<class tipoelem>
void alberonv<tipoelem>::insprimosottoalbero(alberonv<tipoelem> T, typename alberonv<tipoelem>::nodo nod) {
    cout << T.legginodo(T.radice()) << "\n\n";
    int scostamento = nod + 1;
    node *tmp = tree;

    dimtree = nodialbero + T.nodialbero;
    node *albero = new struct Node[dimtree];
    for(int i = 0; i <= nod; i++)
        albero[i] = tree[i];
    for(int i = 0; i < T.nodialbero; i++) {
        albero[nod + 1 + i].etichetta = T.tree[i].etichetta;
        if(i == T.radice())
            albero[nod + 1 + i].padre = nod;
        else
            albero[nod + 1 + i].padre = T.tree[i].padre + scostamento;
    }
    for(int i = nod + 1; i < this->nodialbero; i++) {
        albero[i + T.nodialbero] = tree[i];
        albero[i + T.nodialbero].padre = tree[i].padre + T.nodialbero;
    }
    this->nodialbero += T.nodialbero;
    for(int i = nodialbero; i < dimtree; i++)
        albero[i].padre = -1;
    tree = albero;
    delete tmp;
}

template<class tipoelem>
void alberonv<tipoelem>::inssottoalbero(alberonv<tipoelem> T, typename alberonv<tipoelem>::nodo nod) {
    node *albero = new struct Node[this->nodialbero + T.nodialbero];
    node *tmp = tree;
    dimtree += T.nodialbero;

    for(int i = 0; i <= nod; i++)
        albero[i] = tree[i];
    for(int i = 0; i < T.nodialbero; i++) {

        albero[nod + i + 1].etichetta = T.tree[i].etichetta;
        if(i == T.radice())
            albero[nod + i + 1].padre = tree[nod].padre;
    }
}

```

```

        else
            albero[nod + i + 1].padre = T.tree[i].padre + nod + 1;
    }
    for(int i = nod + 1; i < nodialbero; i++) {
        albero[i + T.nodialbero] = tree[i];
        albero[i + T.nodialbero].padre = tree[i].padre + T.nodialbero;
    }
    tree = albero;
    nodialbero += T.nodialbero;
    for(int i = nodialbero; i < dimtree; i++)
        albero[i].padre = -1;
    delete tmp;
}

template<class tipoelem>
void alberonv<tipoelem>::cancsottoalbero(typename alberonv<tipoelem>::nodo nod) {
    if(!foglia(nod)) {
        nodo v = primofiglio(nod);
        while(!ultimofratello(v)) {
            cancottoalbero(v);
            v = succfratello(v);
        }
        cancottoalbero(v);
    }
    tree[nod].padre = -1;
    nodialbero--;
}

#endif /* NARIOVETTOREPADRI_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.leggilista(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.finelist(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.finelist(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.finelist(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.finelist(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         return trovato;
90     }
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     }
142

```

```

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
170

```

```

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 vettore non ordinato

```

1 // (@Integratore: Alex Spina)
2
3 // Dizionario con vettore non ordinato - Creatore: D'Aversa - Integratore: De Rinaldis
4 #ifndef DIZIONARIO_H_INCLUDED
5 #define DIZIONARIO_H_INCLUDED
6 #include <iostream>
7 #include <cstdlib>
8 #include "Entry.h"
9 #define DIM 1000
10 using namespace std;
11 template<typename K, class A> class Dizionario {
12
13     typedef Entry<K, A> entry;
14
15 public:
16     Dizionario();
17     ~Dizionario();
18     void crea();
19     bool appartiene(const K);
20     void inserisci(K, A);
21     A recupera(const K);
22     void aggiorna(const K, A);
23     void cancella(const K);
24
25 private:
26     unsigned int lunghezza;
27     entry elementi[DIM];
28 };
29
30 template<typename K, typename A>
31 Dizionario<K, A>::Dizionario(){
32
33     crea();
34 }
35
36 template<typename K, typename A>
37 void Dizionario<K, A>::crea(){
38
39     lunghezza = 0;
40 }
41
42 template<typename K, typename A>
43 Dizionario<K, A>::~Dizionario(){
44 }
45
46
47 template<typename K, typename A>
48 bool Dizionario<K, A>::appartiene(const K k){
49
50     bool trovato = false;
51
52     for(int i = 0; i<lunghezza && trovato == false;i++){
53         if(elementi[i].get_key() == k) trovato = true;
54     }
55
56     return trovato;
57 }
58
59 template<typename K, typename A>
60 void Dizionario<K, A>::inserisci(K k, A a){
61
62     elementi[lunghezza].set_key(k);
63     elementi[lunghezza].set_value(a);
64     lunghezza++;
65 }
66
67 template<typename K, typename A>
68 void Dizionario<K, A>::cancella(K k){
69
70     bool trovato = false;
71
72     for(int i = 0; i<lunghezza && trovato == false;i++){
73
74         if(elementi[i].get_key() == k){
75
76             trovato = true;
77
78             for(int j = i; j<lunghezza; j++){
79
80                 elementi[j].set_key(elementi[j+1].get_key());
81                 A h = elementi[j+1].get_value();
82                 elementi[j].set_value(h);
83             }
84
85             lunghezza--;
86         }
87     }
88 }
```

```

87     }
88 }
89
90 template<typename K, typename A>
91 Dizionario<K, A>::recupera(K k) {
92
93     bool trovato = false;
94     int i = -1;
95
96     do{
97         i++;
98         if(elementi[i].get_key() == k) trovato = true;
99     }while(!trovato && i<lunghezza);
100
101     A b = elementi[i].get_value();
102
103     return b;
104 }
105
106
107
108 template<typename K, typename A>
109 void Dizionario<K, A>::aggiorna(K k, A a) {
110
111     bool trovato = false;
112
113     for(int i = 0; i<lunghezza && trovato == false;i++){
114
115         if(elementi[i].get_key() == k){
116
117             trovato = true;
118             elementi[i].set_value(a);
119         }
120     }
121 }
122
123 #endif // DIZIONARIO_H_INCLUDED
124

```

6.8.3 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.legglista(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.legglista(pos).get_key() == chiave){
65             trovato=true;
66             table.canlista(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.finelista(pos) && !trovato){
80         if(table.legglista(pos).get_key() == chiave){
81             valore=table.legglista(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.finelista(pos) && !trovato){
95         if(table.legglista(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

```

6.8.4 Dizionario con array con Lista di trabocco

```

#ifndef DIZIONARIO_H
#define DIZIONARIO_H

/* Dizionario presente nel progetto di Carlucci,
   modificato per essere integrato con la Lista presente nel progetto base di Bagordo e renderlo quindi
   intercambiabile.
   Modifica di Samuele Pesce */

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

using namespace std;

typedef unsigned int HashValue;

template<typename K, class A> class Dizionario { // A=tipo elemento, K=tipo chiave
    typedef Entry<K, A> entry;

public:
    Dizionario();
    ~Dizionario();
    void crea();
    bool appartiene(const K); // restituisce true se l'elemento appartiene al dizionario
    void inserisci(K, A); // inserisce un elemento nel dizionario
    A recupera(K); // restituisce l'elemento corrispondente alla chiave (se esiste)
    void aggiorna(K, A); // aggiorna l'elemento
    void cancella(K); // elimina l'elemento
private:
    unsigned const int maxDim = 101;
    unsigned int H(K); // calcola il valore hash della chiave
    unsigned int lunghezza; // lunghezza massima del dizionario
    unsigned int nElementi; // elementi presenti in un certo istante nel dizionario
    Lista<entry*>* table; // liste di trabocco per le entry //Modifica PMF(agenda)
    HashValue hash(string);
    HashValue hash(int);
};

template<typename K, typename A>
Dizionario<K, A>::Dizionario()
{
    crea();
}

template<typename K, typename A>
Dizionario<K, A>::~Dizionario() {
    delete[] table;
}

template<typename K, typename A>
void Dizionario<K, A>::crea()

```

```

template<typename K, typename A>
void Dizionario<K, A>::crea()
{
    lunghezza = maxDim;
    nelementi = 0;
    table = new Lista<entry>[maxDim]; //Modifica PMF(agenda)
}

template<typename K, typename A>
void Dizionario<K, A>::inserisci(K chiave, A element)
{
    entry e(chiave, element);
    unsigned int h = H(chiave);
    typename Lista<entry>::posizione pos = table[h].primolista();
    table[h].inslista(e, pos);
    nelementi++;
}

template<typename K, typename A>
bool Dizionario<K, A>::appartiene(K key)
{
    typename Lista<entry>::posizione iter; //Lista<entry>::posizione
    bool trovato = false;
    unsigned int pos = H(key);
    iter = table[pos].primolista();
    while (!table[pos].finelista(iter) && !trovato)
    {
        if (key == table[pos].leggilsta(iter).get_key())
            trovato = true;
        iter = table[pos].succlista(iter);
    }
    return trovato;
}

template<typename K, typename A>
A Dizionario<K, A>::recupera(K key)
{
    typename Lista<entry>::posizione iter;
    bool trovato = false;
    A elem;
    unsigned int pos = H(key);
    iter = table[pos].primolista();
    while (!table[pos].finelista(iter) && !trovato) {
        if (key == table[pos].leggilsta(iter).get_key()) {
            trovato = true;
            elem = table[pos].leggilsta(iter).get_value();
        }
        iter = table[pos].succlista(iter);
    }
    return elem;
}

```

```

        .....
```

```

    }

template<typename K, typename A>
void Dizionario<K, A>::aggiorna(K key, A elemento)
{
    cancella(key);
    inserisci(key, elemento);
}

template<typename K, typename A>
void Dizionario<K, A>::cancella(K chiave)
{
    typename Lista<entry>::posizione iter;
    bool trovato = false;
    unsigned int pos = H(chiave);
    iter = table[pos].primolista();
    while (!table[pos].finelista(iter) && !trovato) {
        if (chiave == table[pos].legglista(iter).get_key()) //Modifica PMF(agenda)
            trovato = true;
        else
            iter = table[pos].succlista(iter);
    }
    if (trovato)
    {
        table[pos].canclista(iter);
        nelementi--;
    }
}

template<typename K, typename A>
unsigned int Dizionario<K, A>::H(K chiave) {
    return (hash(chiave) % lunghezza);
}

template<typename K, typename A>
HashValue Dizionario<K, A>::hash(string str)
{
    HashValue hash = 5381;
    int l = str.length();
    for (int i = 0; i < l; i++)
        hash = hash * 33 + str[i];
    return hash;
}

template<typename K, typename A>
HashValue Dizionario<K, A>::hash(int i) {
    return i;
}

#endif // DIZIONARIO_H

```

6.9 GRAFI

6.9.1 Matrice di Adiacenza

```

/**
 * @file Grafo.h
 * Definizione struttura dati Grafo realizzata mediante Matrice di Adiacenza

```

```
* @author Graziano Montanaro.  
* @date Anno Accademico 2018/19.  
* Modificato da Gravina Antonio  
*/
```

```
#ifndef GRAFO_H_INCLUDED  
#define GRAFO_H_INCLUDED  
  
#include "CellaGrafo.h"  
#include "CellaNodo.h"  
#include <iostream>  
#include "Lista.h"  
#define maxNodi 50  
  
  
  
template <class tipoElem, class tipoPeso>  
class Grafo  
{  
public:  
    class nodo;  
    Grafo();  
    ~Grafo();  
    void creagrafo();  
    bool grafovuento() const;  
    void insarco(nodo&,nodo&);  
    void cancarco(nodo&,nodo&);  
    void insnodo(nodo &);  
    void cancnodo(nodo&);  
    Lista<nodo> adiacenti(nodo&) const;  
    bool esistenodo(nodo&) const;  
    bool esistearco(nodo&,nodo&) const;
```

```

void scrivinodo(tipoElem,nodo);
void scriviarco(tipoPeso,nodo&,nodo&);

tipoElem legginodo(nodo&) const;
tipoPeso leggiarco(nodo,nodo) const;

private:

    unsigned int nodiUsati;
    unsigned int primoLibero();
    CellaGrafo<tipoPeso> matrice[maxNodi][maxNodi];
    CellaNodo<tipoElem> nodi[maxNodi];
};

// Definizione del tipo nodo

template <class tipoElem, class tipoPeso>
class Grafo<tipoElem,tipoPeso>::nodo
{
public:
    nodo();
    nodo(const nodo&);

    ~nodo();

    int getNodo();

    void scrivinodo(int n);

    bool operator == ( const nodo& );
    bool operator != ( const nodo& );

private:

```

```

int valore;
};

template <class tipoElem, class tipoPeso>
Grafo<tipoElem , tipoPeso>::Grafo()
{
    nodiUsati = 0;
    for(int i=0;i<maxNodi;i++)
    {
        nodi[i].setPresente(false);
        for(int j=0;j<maxNodi;j++)
            matrice[i][j].setUsato(false);
    }
}

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

template <class tipoElem, class tipoPeso>
bool Grafo<tipoElem , tipoPeso>::grafovuento() const
{
    return nodiUsati == 0;
}

template <class tipoElem, class tipoPeso>
bool Grafo<tipoElem, tipoPeso>::esistenodo(nodo &n) const

```

```

{
    if(n.getNodo()>=0 && n.getNodo()<nodiUsati)
        return nodi[n.getNodo()].getPresente();
    else
        return false;
}

template <class tipoElem, class tipoPeso>
tipoPeso Grafo<tipoElem , tipoPeso>::leggiarco(nodo n1, nodo n2) const
{
    if(esistearco(n1,n2))
        return matrice[n1.getNodo()][n2.getNodo()].getPeso();
}

template <class tipoElem, class tipoPeso>
void Grafo<tipoElem , tipoPeso>::insnodo(nodo & n)
{
    if(!esistenodo(n))
    {
        if(nodiUsati<maxNodi)
        {
            unsigned int i = primoLibero();
            nodi[i].setPresente(true);
            nodi[i].setArchi(0);
            n.scrivinodo(i);
            nodiUsati++;
        }
    }
}

```

```

template <class tipoElem, class tipoPeso>
void Grafo<tipoElem , tipoPeso>::cancnodo(nodo& n)
{
    bool isolato = true;
    for(int i=0;i<nodiUsati;i++ && isolato)
    {
        if(matrice[i][n.getNodo()].getUsato())
            isolato = false;
    }
    for(int i=0;i<nodiUsati;i++ && isolato)
    {
        if(matrice[n.getNodo()][i].getUsato())
            isolato = false;
    }
    if(esistenodo(n) && isolato)
    {
        nodo* temp = new nodo;
        for(int i=0;i<nodiUsati;i++)
        {
            temp->scrivinodo(i);
            if(esistearco(*temp,n))
                cancarco(*temp,n);
            if(esistearco(n,*temp))
                cancarco(n,*temp);
        }
        nodi[n.getNodo()].setPresente(false);
        nodiUsati--;
    }
}

```

```

}

template <class tipoElem, class tipoPeso>
void Grafo<tipoElem , tipoPeso>::insarco(nodo &n1, nodo &n2)
{
    if(!esistearco(n1,n2))
    {
        matrice[n1.getNodo()][n2.getNodo()].setUsato(true);
        nodi[n1.getNodo()].setArchi(nodi[n1.getNodo()].getArchi()+1);
        nodi[n2.getNodo()].setArchi(nodi[n2.getNodo()].getArchi()+1);
    }
}

template <class tipoElem, class tipoPeso>
void Grafo<tipoElem , tipoPeso>::cancarco(nodo& n1, nodo& n2)
{
    if(esistearco(n1,n2))
    {
        matrice[n1.getNodo()][n2.getNodo()].setUsato(false);
        nodi[n1.getNodo()].setArchi(nodi[n2.getNodo()].getArchi() -1);
        nodi[n2.getNodo()].setArchi(nodi[n2.getNodo()].getArchi() -1);
    }
}

template <class tipoElem, class tipoPeso>
void Grafo<tipoElem , tipoPeso>::scrivinodo(tipoElem e, nodo n)
{

```

```

if(esistenodo(n))
{
    nodi[n.getNode()].setEtichetta(e);
}

}

template <class tipoElem, class tipoPeso>
tipoElem Grafo<tipoElem , tipoPeso>::legginodo(nodo& n) const
{
    if(esistenodo(n))
    {
        return nodi[n.getNode()].getEtichetta();
    }
}

template <class tipoElem, class tipoPeso>
void Grafo<tipoElem , tipoPeso>::scriviarco(tipoPeso p,nodo& n1, nodo& n2)
{
    if(!esistearco(n1,n2))
    {
        matrice[n1.getNode()][n2.getNode()].setPeso(p);
    }
}

template <class tipoElem, class tipoPeso>
bool Grafo<tipoElem , tipoPeso>::esistearco(nodo& n1, nodo& n2) const
{
    if(esistenodo(n1))
        if(esistenodo(n2))

```

```

        return matrice[n1.getNodo()][n2.getNodo()].getUsato();

    }

template <class tipoElem, class tipoPeso>
Lista<typename Grafo<tipoElem,tipoPeso>::nodo> Grafo<tipoElem , tipoPeso>::adiacenti(nodo& n)
const
{
    Lista<nodo> l;
    typename Lista<nodo>::posizione p = l.primolista();
    for(int i=0;i<nodiUsati;i++)
    {
        if(matrice[n.getNodo()][i].getUsato())
        {
            nodo* temp = new nodo();
            temp->scrivinodo(i);
            l.inslista(*temp,p);
        }
    }
    // Nodi di cui n è coda
    /*for(int i=0;i<nodiUsati;i++)
    {
        if(matrice[i][n].getUsato())
            if(!matrice[n][i].getUsato())
                l.inslista(i,p);
    }*/
    return l;
}

template <class tipoElem, class tipoPeso>
unsigned int Grafo<tipoElem,tipoPeso>::primoLibero()

```

```
{  
    nodo i;  
  
    i.scrivinodo(-1);  
  
    bool libero=false;  
  
    while(!libero)  
  
    {  
  
        i.scrivinodo(i.getNodo()+1);  
  
        if(nodi[i.getNodo()].getPresente()==false)  
            libero = true;  
  
    }  
  
    return i.getNodo();  
}
```

//Operatori NODO

```
template <class tipoElem, class tipoPeso>  
Grafo<tipoElem,tipoPeso>::nodo::nodo()  
{  
    valore = -1;  
}
```

```
template <class tipoElem, class tipoPeso>  
Grafo<tipoElem,tipoPeso>::nodo::nodo(const nodo& n)  
{  
    valore = n.valore;  
}
```

```

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

}

template <class tipoElem, class tipoPeso>
int Grafo<tipoElem, tipoPeso>::nodo::getNodo()
{
    return valore;
}

template <class tipoElem, class tipoPeso>
void Grafo<tipoElem, tipoPeso>::nodo::scrivinodo(int n)
{
    valore = n;
}

template <class tipoElem, class tipoPeso>
bool Grafo<tipoElem, tipoPeso>::nodo:: operator == ( const nodo& n)
{
    return valore == n.valore;
}

template <class tipoElem, class tipoPeso>
bool Grafo<tipoElem, tipoPeso>::nodo:: operator != ( const nodo& n)
{
    return valore != n.valore;
}

```

```
#endif // GRAFO_H_INCLUDED
```

6.9.1.1 *CellaGrafo.h*

```
/**  
 * @file CellaGrafo.h  
 * Definizione della Cella presente nella matrice di Grafo.h ( matrice di adiacenza)  
 * @author Graziano Montanaro.  
 * @date Anno Accademico 2018/19.  
 * Modificato da Gravina Antonio  
 */
```

```
#ifndef CELLAGRAFO_H_INCLUDED
```

```
#define CELLAGRAFO_H_INCLUDED
```

```
template <class tipoPeso>  
  
class CellaGrafo  
{  
  
public:  
  
    CellaGrafo();  
  
    CellaGrafo(const CellaGrafo<tipoPeso>& );  
  
    ~CellaGrafo();  
  
    void setUsato(bool);  
  
    void setPeso(tipoPeso);  
  
    bool getUsato() const;  
  
    tipoPeso getPeso() const;  
  
    bool operator == ( const CellaGrafo<tipoPeso> & );  
  
    bool operator != ( const CellaGrafo<tipoPeso> & );
```

```
void operator = ( const CellaGrafo<tipoPeso> & );  
private:  
    bool usato;  
    tipoPeso peso;  
};
```

```
template <class tipoPeso>  
CellaGrafo<tipoPeso>::CellaGrafo()  
{  
    usato = false;  
    peso = tipoPeso();  
}
```

```
template <class tipoPeso>  
CellaGrafo<tipoPeso>::CellaGrafo(const CellaGrafo<tipoPeso>& c)  
{  
    usato = c.usato;  
    peso = c.peso;  
}
```

```
template <class tipoPeso>  
CellaGrafo<tipoPeso>::~CellaGrafo()  
{  
}
```

```
template <class tipoPeso>  
void CellaGrafo<tipoPeso>::setUsato(bool b)  
{  
    usato = b;  
}
```

```

template <class tipoPeso>
void CellaGrafo<tipoPeso>::setPeso(tipoPeso p)
{
    peso = p;
}

template <class tipoPeso>
bool CellaGrafo<tipoPeso>::getUsato() const
{
    return usato;
}

template <class tipoPeso>
tipoPeso CellaGrafo<tipoPeso>::getPeso() const
{
    return peso;
}

template <class tipoPeso>
bool CellaGrafo<tipoPeso>::operator == (const CellaGrafo<tipoPeso> & c)
{
    return peso == c.peso;
}

template <class tipoPeso>
bool CellaGrafo<tipoPeso>::operator != (const CellaGrafo<tipoPeso> & c)
{
    return peso != c.peso;
}

```

```

template <class tipoPeso>
void CellaGrafo<tipoPeso>::operator = (const CellaGrafo<tipoPeso> & c)
{
    peso = c.peso;
    usato = c.usato;
}

#endif // CELLAGRAFO_H_INCLUDED

```

6.9.1.2 *CellaNodo.h*

```

/**
 * @file CellaNodo.h
 * Definizione della Cella presente nel vettore di Nodi di Grafo.h ( matrice di adiacenza)
 * @author Graziano Montanaro.
 * @date Anno Accademico 2018/19.
 * Modificato da Gravina Antonio
 */

```

```

#ifndef CellaNodo_H_INCLUDED
#define CellaNodo_H_INCLUDED

```

```

template <class tipoElem>
class CellaNodo
{
public:
    CellaNodo();
    CellaNodo(const CellaNodo<tipoElem>&);
    ~CellaNodo();
    void setPresente(bool);
    void setEtichetta(tipoElem );
    void setArchi(int);

```

```
void setIndice(unsigned int);
unsigned int getIndice() const;
bool getPresente() const;
tipoElem getEtichetta() const;
int getArchi() const;
bool operator == (const CellaNodo<tipoElem>&);
bool operator != (const CellaNodo<tipoElem>&);
void operator = (const CellaNodo<tipoElem>&);
```

private:

```
bool presente;
tipoElem etichetta;
int archi;
unsigned int indice;
};
```

```
template <class tipoElem>
CellaNodo<tipoElem>::CellaNodo()
{
    presente = false;
    etichetta = tipoElem();
    archi = 0;
    indice = 0;
}
```

```
template <class tipoElem>
CellaNodo<tipoElem>::CellaNodo(const CellaNodo<tipoElem>& n)
{
    presente = n.presente;
    etichetta = n.etichetta;
    archi = n.archi;
```

```
indice = n.indice;  
}  
  
template <class tipoElem>  
CellaNodo<tipoElem>::~CellaNodo()  
{  
  
}  
  
template <class tipoElem>  
void CellaNodo<tipoElem>::setPresente(bool b)  
{  
    presente = b;  
}  
  
template <class tipoElem>  
void CellaNodo<tipoElem>::setEtichetta(tipoElem e)  
{  
    etichetta = e;  
}  
  
template <class tipoElem>  
void CellaNodo<tipoElem>::setArchi(int a)  
{  
    archi = a;  
}  
  
template <class tipoElem>  
void CellaNodo<tipoElem>::setIndice(unsigned int a)  
{
```

```

indice = a;
}

template <class tipoElem>
unsigned int CellaNodo<tipoElem>::getIndice() const
{
    return indice;
}

template <class tipoElem>
tipoElem CellaNodo<tipoElem>::getEtichetta() const
{
    return etichetta;
}

template <class tipoElem>
bool CellaNodo<tipoElem>::getPresente() const
{
    return presente;
}

template <class tipoElem>
int CellaNodo<tipoElem>::getArchi() const
{
    return archi;
}

template <class tipoElem>
bool CellaNodo<tipoElem>:: operator ==(const CellaNodo<tipoElem>& n)

```

```

{
    return etichetta == n.etichetta;
}

template <class tipoElem>
bool CellaNodo<tipoElem>:: operator !=(const CellaNodo<tipoElem>& n)
{
    return etichetta != n.etichetta;
}

template <class tipoElem>
void CellaNodo<tipoElem>:: operator = (const CellaNodo<tipoElem>& n)
{
    presente = n.presente;
    etichetta = n.etichetta;
    archi = n.archi;
    indice = n.indice;
}

#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)
   e 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
84.         tutti nodi
85.     {
86.         table[i].esiste=false;
87.     }
88.
89. template<class tipoElem,class tipoPeso>
90. bool Grafo<tipoElem, tipoPeso>::grafovuto() const //restituisce true se il grafo è vuoto,
91.     false altrimenti
92. {
93.     return (nelementi==0);
94.
95. template<class tipoElem, class tipoPeso>
96. void Grafo<tipoElem, tipoPeso>::insnodo(nodo &n) //inserisce il nodo che sarà identificato
97.     dall'indice n
98. {
99.     //passaggio del parametro per indirizzo perchè la variabile sarà modificata
100.    if (!esistenodo(n) && nelementi<1000) // precondizione nodo non appartenente
101.        {
102.            n=primolibero(); //la posizione in cui metterò il nodo del vettore sarà la
103.            prima libera
104.            table[n].esiste=true; //setto a true il suo campo esiste
105.            nelementi++; //aumento il contatore di nodi del grafo
106.        }
107.    template<class tipoElem, class tipoPeso>
108.    void Grafo<tipoElem, tipoPeso>::insarco(nodo n,nodo m) //inserisce l'arco che esce d
109.        al nodo n ed entra in m
110.    {
111.        if (esistenodo(n) && esistenodo(m) && !esistearco(n,m))//precondizione nodi app
112.            artenenti e arco non esistente
113.            {
114.                typename Lista<adiacente>::posizione indice=table[n].adiac.primolista();
115.                adiacente temp; //creo l'elemento da inserire
116.                temp.scrivinodo(m); //setto l'adiacente
117.                table[n].adiac.inslista(temp,indice); //inserisco il nodo m negli adiacenti
118.                    di n (in prima posizione)
119.            }
120.        template<class tipoElem, class tipoPeso>
121.        void Grafo<tipoElem, tipoPeso>::cancnodo(nodo n) //elimina il nodo n
122.        {
123.            if (esistenodo(n))// 1)-----
124.            {
125.                //|
126.                if (table[n].adiac.listavuota())// 2)      //|
127.                {
128.                    //|
129.                    bool libero=true; //|
130.                    int i=0; //|
131.                    while (i<maxnodi && libero) //|
132.                    {
133.                        //|--
134.                        - precondizione : nodo esistente 1),che non ha archi uscenti 2) nè entranti 3)
135.                        if (esistenodo(i)) //|
136.                        {
137.                            //|
138.                            libero=!esistearco(i,n); //|
139.                            i++; //|
140.                        } //|
141.                    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.legglista(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.     List<typename Grafo<tipoElem,tipoPeso>::nodo> Grafo<tipoElem,tipoPeso>::adiacenti(nodo n) const //restituisce la lista di adiacenti di n
171.     {
172.         List<nodo> lista; //lista da restituire
173.         adiacente temp; //comodo
174.         if (esistenodo(n)) // precondizione nodo appartenente al grafo
175.         {
176.             typename Lista<adiacente>::posizione indice=table[n].adiac.primolista(); //indice di scansione della lista di adiacenti (lista di adiacente)
177.             typename Lista<nodo>::posizione indice2=lista.primolista(); //indice di scansione della lista di adiacendi da restituire (lista di nodo)
178.             while (!table[n].adiac.finelista(indice)) //scansione della prima lista
179.             {
180.                 temp=table[n].adiac.legglista(indice); //lettura di adiacente
181.                 lista.inslista(temp.legginodo(),indice2); //inserisco (in coda) nella lista da restituire solo il riferimento al nodo adiacente (senza il peso dell'arco)
182.                 indice=table[n].adiac.succlista(indice);
183.                 indice2=lista.succlista(indice2);
184.             }
185.         }
186.         return(lista);
187.     }
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.finlista(indice) && !esiste) //scandisco la lista finché non trovo l'elemento o è finita
204.                  {
205.                      if (table[n].adiac.legglist(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>::leggino(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.finlista(indice) && !aggiornato) //scandisco la lista finché non trovo l'elemento o è finita
238.                  {
239.                      if (table[n].adiac.legglist(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 dell'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 appartenenente 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.finellista(indice) && !letto) //scandisco la lista fi
        nchè non trovo l'elemento o è finita
259.             {
260.                 if (table[n].adiac.legglist(indice).leggnodo()==m)
261.                 {
262.                     a=table[n].adiac.legglist(indice).leggipeso();
263.                     letto=true;
264.                 }
265.                 indice=table[n].adiac.succlista(indice);
266.             }
267.         }
268.         return(a);
269.     }
270.
271.
272.     //operatori ausiliare
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. //sovraffaccio output
73.
```

```

74. template<class nodo,class tipoPeso> ostream& operator<<(ostream& os, const Adiacenza<nodo,
75. do, tipoPeso>& a)
76. {
77.     os<<"(";<<a.legginodo()<<"|"<<a.leggipeso()<<")";
78. }
79.
80. #endif

```

6.10 ALBERI BINARI

6.10.1 REALIZZAZIONE CON CURSORI DELL'ALBERO BINARIO

//Implementato da Joseph Tortorella
//Realizzazione con cursori dell'albero binario

```

#ifndef BINALBERO_CURSORI_H_
#define BINALBERO_CURSORI_H_

template <class T> class BinAlbero {
    static const int NIL = -1;

    public:

    // tipi
    typedef T tipoelem;
    typedef int Nodo;
    struct _cella{
        int genitore;
        int sinistro;
        int destro;
        tipoelem valore;
    };
    typedef struct _cella Cella;

    // costruttori e distruttori
    BinAlbero();
    BinAlbero(int);
    ~BinAlbero();

    // operatori
    void creaBinAlbero();
    bool binAlberoVuoto();
    Nodo binRadice();
    Nodo binPadre(Nodo);
    Nodo figlioSinistro(Nodo);
    Nodo figlioDestro(Nodo);
    bool sinistroVuoto(Nodo);
    bool destroVuoto(Nodo);
    void costrBinAlbero(BinAlbero<T>, BinAlbero<T>);
    void cancSottoBinAlbero(Nodo);
    T leggiNodo(Nodo);
    void scriviNodo(Nodo , tipoelem );
    void insBinRadice(Nodo);
    bool insFiglioSinistro(Nodo);
    bool insFiglioDestro(Nodo);

    private:
    int MAXLUNG;
    Cella *spazio;

```

```

        int nNodi;
        Nodo inizio;
        Nodo libera;
    };

    template <class T>
    BinAlbero<T>::BinAlbero(){
        MAXLUNG = 100;
        spazio = new Cella[MAXLUNG];
        creaBinAlbero();
    }

    template <class T>
    BinAlbero<T>::BinAlbero(int nNodi): MAXLUNG(nNodi) {
        spazio = new Cella[nNodi];
        creaBinAlbero();
    }

    template <class T>
    BinAlbero<T>::~BinAlbero() {
        cancSottoBinAlbero(inizio);
        delete[] spazio;
    }

    template <class T>
    void BinAlbero<T>::creaBinAlbero() {
        inizio = NIL;
        for (int i = 0; i < MAXLUNG; i++) {
            spazio[i].sinistro = (i+1) % MAXLUNG;
        }
        libera = 0;
        nNodi = 0;
    }

    template <class T>
    bool BinAlbero<T>::binAlberoVuoto() {
        return(nNodi == 0);
    }

    template <class T> typename BinAlbero<T>::Nodo
    BinAlbero<T>::binRadice() {
        return(inizio);
    }

    template <class T> typename BinAlbero<T>::Nodo
    BinAlbero<T>::binPadre(Nodo n) {
        if (n != inizio)
            return (spazio[n].genitore);
        else return(n);
    }

    template <class T> typename BinAlbero<T>::Nodo
    BinAlbero<T>::figlioSinistro(Nodo n) {
        if (!sinistroVuoto(n))
            return (spazio[n].sinistro);
        else return(n);
    }

    template <class T> typename BinAlbero<T>::Nodo

```

```

BinAlbero<T>::figlioDestro(Nodo n) {

    if (!destroVuoto(n))
        return (spazio[n].destro);
    else return(n);
}

template <class T>
bool BinAlbero<T>::sinistroVuoto(BinAlbero<T>::Nodo n) {
    return (spazio[n].sinistro == NIL);
}

template <class T>
bool BinAlbero<T>::destroVuoto(BinAlbero<T>::Nodo n) {
    return (spazio[n].destro == NIL);
}

template <class T>
void BinAlbero<T>::insBinRadice(BinAlbero<T>::Nodo n) {
    if (inizio == NIL) {
        inizio = libera;
        libera = spazio[libera].sinistro;
        spazio[inizio].genitore = NIL;
        spazio[inizio].sinistro = NIL;
        spazio[inizio].destro = NIL;
        nNodi++;
    }
}

template <class T>
void BinAlbero<T>::cancSottoBinAlbero(Nodo n) {
    if (n != NIL) {
        if (!sinistroVuoto(n))
            cancSottoBinAlbero(sazio[n].sinistro);
        if (!destroVuoto(n))
            cancSottoBinAlbero(sazio[n].destro);
        if (n != inizio) {
            Nodo p = binPadre(n);

            if (spazio[p].sinistro == n)
                spazio[p].sinistro = NIL; //se il nodo da cancellare è figlio
sinistro
            else
                spazio[p].destro = NIL; //se il nodo da cancellare è figlio
destro
        }
        else
            inizio = NIL; //se il nodo da cancellare è radice
        nNodi--;
        spazio[n].sinistro = libera;
        libera = n;
    }
}

template <class T>
T BinAlbero<T>::leggiNodo(Nodo n) {
    if (n != NIL)
        return (spazio[n].valore);
}

```

```

template <class T>
void BinAlbero<T>::scriviNodo(Nodo n, tipoelem a) {

    if (n != NIL)
        spazio[n].valore = a;
}

#endif /* BINALBERO_CURSORI_H_ */

```

6.10.2 REALIZZAZIONE CON VETTORE

```

1  #ifndef BINALBERO_H_INCLUDED
2  #define BINALBERO_H_INCLUDED
3  #include "Nodo_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         tipoelem<Cella_Binbero TIPOETICHETTA>* Nodo;
16         //DICHIARAZIONE NODO NULLO
17         static const elem<Nodo> nil=empty();
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 bipadre(Nodo) const;
30         //METODO CHE RESTITUISCE VERO SE IL FIGLIO SINISTRO DEL NODO PASSATO NON ESISTE, FALSO ALTRIMENTI
31         bool sinistrovuoto(Nodo) const;

```

```

32         //METODO CHE RESTITUISCE VERO SE IL FIGLIO DESTRO DEL NODO PASSATO NON ESISTE, FALSO ALTRIMENTI
33         bool destrovuoto(Nodo) const;
34         //METODO CHE RESTITUISCE IL FIGLIO SINISTRO DEL NODO PASSATO
35         Nodo figliosinistro(Nodo) const;
36         //METODO CHE RESTITUISCE IL FIGLIO DESTRO DEL NODO PASSATO
37         Nodo figliodestruo(Nodo) const;
38         //METODO CHE DATI DUE ALBERI, CREA UNA RADICE NULLA ALL'ALBERO BINARIO IMPLICITO,
39         //INSERISCE COME FIGLIO SINISTRO LA RADICE DEL PRIMO ALBERO PASSATO E COPIATI I SUOI DISCENDENTI,
40         //E INSERISCE COME FIGLIO DESTRO IL SECONDO ALBERO PASSATO E COPIATI I SUOI DISCENDENTI.
41         void costrbinalbero(BinAlbero<TIPOETICHETTA> &, BinAlbero<TIPOETICHETTA> &);
42         //METODO CHE PASSATO UN NODO LO CANCELLA E CANCELLA TUTTI I SUOI DISCENDENTI
43         void cancottobinalbero(Nodo);
44         //METODO CHE PASSATO UN NODO RESTITUISCE LA SUA ETICHETTA
45         TIPOETICHETTA legginodo(Nodo) const;
46         //METODO CHE PASSATO UN NODO SCRIVE AL SUO INTERNO L'ETICHETTA DI TIPO TIPOELEM
47         void scrivinodo(Nodo, TIPOETICHETTA);
48         //METODO CHE INSERISCE LA RADICE NULLA
49         void insbinradice();
50         //METODO CHE INSERISCE COME FIGLIO SINISTRO IL NODO PASSATO
51         void insfigliosinistro(Nodo);
52         //METODO CHE INSERISCE COME FIGLIO DESTRO IL NODO PASSATO
53         void insfigliodestruo(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             cancsottobinalbero(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         return radice;
89     }
90
91     template <class TIPOETICHETTA>
92     typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binpadre(Nodo padre, const {
93         return padre->getPadre();
94     }
```

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
33     TIPOETICHETTA getEtichetta();
34     //OVERLOAD DELL'OPERATORE ==
35     bool operator == (Cella_Binalbero<TIPOETICHETTA> );
36     //COSTRUTTORE DI COPIA
37     Cella_Binalbero(const Cella_Binalbero& );
38     //OVERLOAD DELL'OPERATORE =
39     void operator=(const Cella_Binalbero& );
40     private:
41         Cella_Binalbero<TIPOETICHETTA>* FiglioDx;
42         Cella_Binalbero<TIPOETICHETTA>* FiglioSx;
43         Cella_Binalbero<TIPOETICHETTA>* Padre;
44         TIPOETICHETTA etichetta;
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         FiglioSx=sx;
61     }
62 }
```

```

63     < TIPOETICHETTA>
64     Cella_Binalbero<TIPOETICHETTA>::setFiglioDx(Cella_Binalbero<TIPOETICHETTA>* dx) {
65         FiglioDx=dx;
66     }
67
68     < TIPOETICHETTA>
69     Cella_Binalbero<TIPOETICHETTA>::setPadre(Cella_Binalbero<TIPOETICHETTA>* p) {
70         Padre=p;
71     }
72
73     < TIPOETICHETTA>
74     Cella_Binalbero<TIPOETICHETTA>::setEtichetta(TIPOETICHETTA* e) {
75         etichetta=e;
76     }
77
78     < TIPOETICHETTA>
79     Cella_Dinalbero<TIPOETICHETTA>* Cella_Dinalbero<TIPOETICHETTA>::getFiglioSx() {
80         return FiglioSx;
81     }
82
83     < TIPOETICHETTA>
84     Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getFiglioDx() {
85         return FiglioDx;
86     }
87
88     < TIPOETICHETTA>
89     Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::getPadre() {
90         return Padre;
91     }
92
93     < TIPOETICHETTA>
94     TIPOETICHETTA Cella_Binalbero<TIPOETICHETTA>::getEtichetta() {
95         return etichetta;
96     }
97
98     template <class TIPOETICHETTA>
99     void Cella_Binalbero<TIPOETICHETTA>::operator == (Cella_Binalbero<TIPOETICHETTA>* b) {
100         if (getEtichetta() == b.getEtichetta())
101             return true;
102         else
103             return false;
104     }
105
106     template <class TIPOETICHETTA>
107     Cella_Binalbero<TIPOETICHETTA>* Cella_Binalbero<TIPOETICHETTA>::Cella_Binalbero(Cella_Binalbero<TIPOETICHETTA>* c) {
108         etichetta=c.getEtichetta();
109         FiglioDx=c.getFiglioDx();
110         FiglioSx=c.getFiglioSx();
111         Padre=c.getPadre();
112     }
113
114     template <class TIPOETICHETTA>
115     void Cella_Binalbero<TIPOETICHETTA>::operator != (Cella_Binalbero<TIPOETICHETTA>* c) {
116         etichetta!=c.getEtichetta();
117         Padre!=c.getPadre();
118         FiglioDx!=c.getFiglioDx();
119         FiglioSx!=c.getFiglioSx();
120     }
121 #endif // NODO_ALBERO_BINARIO_H_INCLUDED

```

```

94
95
96     template <class TIPOETICHETTA>
97     bool BinAlbero<TIPOETICHETTA>::sinistrovuoto(Nodo sx) const {
98         if (sx->getFiglioSx()==nil)
99             return true;
100        else
101            return false;
102    }
103
104    template <class TIPOETICHETTA>
105    bool BinAlbero<TIPOETICHETTA>::destruovoto(Nodo dx) const {
106        if (dx->getFiglioDx()==nil)
107            return true;
108        else
109            return false;
110    }
111
112    template <class TIPOETICHETTA>
113    typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figlosinistro(Nodo sx) const {
114        return sx->getFiglioSx();
115    }
116
117    template <class TIPOETICHETTA>
118    typename BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliodestro(Nodo dx) const {
119        return dx->getFiglioDx();
120    }
121
122    template <class TIPOETICHETTA>
123    void BinAlbero<TIPOETICHETTA>::costrbinalbero(BinAlbero<TIPOETICHETTA>* A, BinAlbero<TIPOETICHETTA>* B) {
124        radice=A;

```

```

125     radice->setPadre(nil);
126
127     if (!A.binalberovuoto()) {
128         radice->setFiglioSx(A.binradice());
129         radice->getFiglioSx()->setPadre(radice);
130     }
131     else
132         radice->setFiglioSx(nil);
133
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             if (temp->getFiglioSx()==r)
157                 temp->setFiglioSx(nil);
158             else
159                 temp->setFiglioDx(nil);
160         }
161         else
162             radice=nil;
163
164         delete r;
165
166     }
167
168
169 template <class TIPOETICHETTA>
170 TIPOETICHETTA BinAlbero<TIPOETICHETTA>::legginodo(Nodo n) const {
171     return n->getEtichetta();
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

```

```

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
206

```

7 TABELLA DI TEST DELLE FUNZIONALITÀ

N.	LUOGO	FUNZIONALITÀ	TEST	PROBLEMA
1	Museo	Percorsi museo	Con successo	Nessuno
2	Cabina di pilotaggio	Guida	Senza successo	Non stampa il contenuto della guida
3	Armadietto della Tua cabina	Rubrica	Senza successo	Non si riesce ad interagire con la rubrica, e le risposte del gioco non sono coerenti con le azioni
4	Ufficio / Deposito	Elenco documenti	Con successo	Nessuno
5	Stadio, Cinema, Mostra d'Arte	Luoghi a pagamento	Con successo	Nessuno
6	In ogni luogo	Navigatore esteso	Senza successo	Non è possibile utilizzare il navigatore
7	Auditorium	Jukebox	Con successo	Nessuno
8	Auditorium	Pannello di controllo luci	Con successo	Nessuno
9	Auditorium	Playlist	Con successo	Nessuno
10	Ristorante / Fast Food	Zaino termico	Con successo	Nessuno
11	Pronto soccorso	Cura	Con successo	Nessuno
12	In ogni luogo	Undo	Con successo	Nessuno
13	In ogni luogo	Stati parziali	Senza successo	I comandi Save e Load non funzionano
14	Poppa dell'astronave, Cabina secondo pilota	Cambio mappa	Con successo	Nessuno
15	Sala giochi	Slot machine	Con successo	Nessuno
16	Stazione	Biglietto	Con successo	Senza il portafoglio è possibile comunque acquistare i biglietti
17	-	Estensione capacità luoghi-verbi-oggetti	Con successo	Nessuno
18	In ogni luogo	Luogo di provenienza	Con successo	Nessuno
19	In ogni luogo	Bacheca	Con successo	Nessuno

N.	LUOGO	FUNZIONALITÀ	TEST	PROBLEMA
20	Meccanico	Visualizza persone in attesa	Con successo	Nessuno
21	Meccanico	Gestione batterie	Con successo	Nessuno
22	Meccanico	Diario meccanico	Con successo	Nessuno
23	Cabina secondo pilota	Maechina del tempo	Con successo	Nessuno
24	Dove sono presenti i personaggi	Dialoghi	Con successo	Nessuno
25	Archivio	Scaricare e Leggere file dal computer	Con successo	Nessuno
26	Scuola	Risolvere cruciverba, rispondere alle maestre	Con successo	Nessuno
27	Tua cabina	Agenda	Senza successo	Manca il file codici.txt contenente i codici degli appunti
28	Ufficio postale	Invio/ritiro pacchi e messaggi	Senza successo	Non è possibile effettuare le operazioni allo sportello perché non riconosce il documento d'identità presente nell'inventario
29	Ogni luogo	Premi	Con successo	Nessuno
30	Sala scommesse	Sala scommesse	Con successo	Nessuno
31	Biblioteca	Libri in prestito	Con successo	Nessuno
32	Palestra	Crea scheda, usa panca, usa tapis	Con successo	Nessuno
33	Ogni luogo	Enigmi	Con successo	Nessuno
34	Aula	Leggi libri	Con successo	Nessuno
35	Ufficio/Deposito, Cabina secondo pilota	Ruba	Con successo	Nessuno
36	Ogni luogo	Riassunto storia	Con successo	Nessuno
37	Ogni luogo	Denaro/portafoglio	Con successo	Nessuno
38	Negozi	Acquisto articoli	Con successo	Nessuno
39	Banca	Cassetta di sicurezza	Senza successo	Non deposita gli oggetti all'interno della cassetta
40	Banca	Preleva/Deposita	Con successo	Nessuno
41	Stazione di servizio	Autobus	Con successo	Nessuno
42	Chiesa	Prega	Con successo	Nessuno
43	Sali&Tabacchi	Leggere	Con successo	Nessuno
44	Sali&Tabacchi	Compra	Con successo	Nessuno
45	Sali&Tabacchi	Prendere	Con successo	Nessuno
46	Sali&Tabacchi	Guarda	Con successo	Nessuno
47	-	Distanza/Sforzo	Con successo	Nessuno

48	Cassaforte	Deposita oggetto nella cassetta	Con successo	Nessuno
49	Cassaforte	Controlla cassetta	Con successo	Nessuno
50	Cassaforte	Ritira oggetto dalla cassetta	Con successo	Nessuno
51	-	Teletrasporto su nuove mappe	Con successo	Nessuno
53	Nel pronto soccorso	Riconoscimento Tessera	Con successo	La tassera viene riconosciuta
54	Laboratorio analisi	Analisi diabete	Con successo	Nessuno
55	Laboratorio analisi	Analisi colesterolo	Con successo	Nessuno
56	Laboratorio analisi	Chiusura droide medico	Con successo	Nessuno

8 PORTING DA WINDOWS A LINUX

Il porting da Windows a Linux, o viceversa, è stato abbandonato in favore di un'unica versione universale in grado di convivere su entrambi i S.O.

Per questo motivo, è stato creato il file “**porting_universale.h**” (e rispettivo .cpp) ed è stato incluso nel main.

I processi per rimuovere le distinzioni tra le due versioni sono stati:

- 1) Rimuovere dai file l’inclusione del file header <windows.h> contenente le dichiarazioni per le funzioni delle Windows Api.
 - a. Sono stati modificati: Astro.cpp, Bacheca.h, Mappa.h, Palestra.h, Gioco.cpp, main.cpp
- 2) Rimuovere dai file l’inclusione del file header <unistd.h> contenente le istruzioni per permettere di eseguire la funzione sleep(...) su sistemi Linux.
 - a. Sono stati modificati: Astro.cpp, Gioco.cpp, Interfaccia.cpp
- 3) Creazione della funzione SLEEP(X) che: nei sistemi Windows, utilizza Sleep(X); nei sistemi Linux, utilizza sleep(X/1000). Rimozione di queste due funzioni dal progetto.
 - a. Sono stati modificati: Astro.cpp, Gioco.cpp, Palestra.cpp
 - b. Attenzione: Su Windows questo parametro X indica millesimi di secondo; su Linux indica secondi, ecco il perché della divisione.
- 4) Creazione della funzione CLEARSCREEN() che: nei sistemi Windows, utilizza system(“cls”); nei sistemi Linux, utilizza system(“clear”). Rimozione di queste due funzioni dal progetto.
 - a. Sono stati modificati i file: Astro.cpp, Gioco.cpp, Interfaccia.cpp
- 5) Aggiustare il main per renderlo automaticamente adattabile alla macchina sulla quale viene eseguito.

8.1 CODIFICA

I. porting_universale.h

```
©Samuele Pesce
File .h per permettere allo stesso progetto di funzionare sia su macchine Windows che su Linux. ( Non testato su macchine Apple ).

#ifndef PORTING_UNIVERSALE_H
#define PORTING_UNIVERSALE_H

#if defined(_WIN32) || defined(_WIN64)
#include <windows.h>
#define stoVenendoEseguitoSuMacchinaWindows
#endif // defined

#if defined(_unix_)
#include <unistd.h>
#include <iostream>
#define stoVenendoEseguitoSuMacchinaLinux
#endif // defined

void CLEARSCREEN();
void SLEEP(int);

#endif // PORTING_UNIVERSALE_H
```

Le costanti simboliche "stoVenendoEseguitoSuMacchinaWindows" e "stoVenendoEseguitoSuMacchinaLinux" sono state pensate per renderle facilmente dinamici altri file, come il main.

II. porting_universale.cpp

```
/* $Samuele Pesce
   File .cpp che implementa le funzioni per permettere allo stesso progetto di funzionare sia su macchine Windows che su Linux. ( Non testato su macchine Apple ). */

#include "porting_universale.h"

#if defined(_WIN32) || defined(_WIN64)
void CLEARSCREEN() { system("cls"); }
void SLEEP(int X) { Sleep(X); }
#endif // defined

#if defined( _unix_ )
void CLEARSCREEN() { system("clear"); }
void SLEEP(int X) { sleep(X/1000); }
#endif // defined

|
```

III. main.cpp

```
#include <cstdlib>
#include <iostream>
#include "Mappa.h"
#include "Astro.h"
#include "porting_universale.h"
using namespace std;

int main(int argc, char *argv[])
{
    #ifdef stoVenendoEseguitoSuMacchinaWindows
    system("title Adventure with Simunav");      //imposto header della finestra dos
    // system("mode con line=100");
    cout<<"[ E' stata rilevata una macchina su S.O Windows ]"<<endl;
    #endif // defined

    #ifdef stoVenendoEseguitoSuMacchinaLinux
    cout<<"[ E' stata rilevata una macchina su S.O Linux ]"<<endl;
    std::cout << "\033[0;" << "Adventure with Simunav" << "\007"; // Permette di inserire il titolo all'applicazione su Linux -- Modifica di Samuele Pesce
    #endif // stoVenendoEseguitoSuMacchinaLinux

    fstream crono("Cronologia.odb",fstream::in); //apro il file su cui è presente la cronologia

    ifstream f("Mappa.nav",ios::in);           //apro il file su cui ho rappresentato la mappa
    if (f.good())                                //se il file di input è stato aperto correttamente
    {
        Mappa M(f,crono); //istanzio la mappa
        Astro partita = Astro();
        partita.ciaK(M,1);

    }
    else cout<<"Errore apertura file \"Mappa.nav\""

```

1) TESTING

Il programma è stato testato su Windows 10 e su Linux distro Ubuntu 20.04 LTS sotto WLS(Windows Linux Subsystem) 2 e testato con macchina virtuale. Stesso file su entrambi i S.O:

Windows 10:

```

5  #include "porting_universale.h"
6  using namespace std;
7
8  int main(int argc, char *argv[])
9  {
10     #ifdef stoVenendoEseguitoSuMacchinaWindows
11         system("title Adventure with Simunav")
12         // system("mode con line=100");
13         cout<<"[ E' stata rilevata una macchina su S.O Windows ]"
14     #endif // defined
15
16     #ifdef stoVenendoEseguitoSuMacchinaLinux
17         cout<<"[ E' stata rilevata una macchina su S.O Linux ]"
18         std::cout << "\033]0;" << "Adventure with Simunav"
19     #endif // stoVenendoEseguitoSuMacchinaLinux
20
21     fstream crono("Cronologia.odb",fstream::out);
22
23     ifstream f("Mappa.nav",ios::in);
24     if (f.good())
25     {
26         Mappa M(f,crono); //istanzio la mappa
27         Astro partita = Astro();
28         partita.ciak(M,f);
29
30     }
31     else cout<<"Errore apertura file \\"Mappa.nav\" - Premi <invio> per continuare -"
32
33     f.close();
34
35     system("PAUSE");
36     return EXIT_SUCCESS;
37 }

```

Adventure with Simunav
[E' stata rilevata una macchina su S.O Windows]
Un attimo di pazienza...
L'astronave condannata:un'avventura #
di Enrico Colombini e Chiara Tovena #
(c) Dinosoft e Jackson Editrice 1985
Sdraiato nella sabbia, ti godi il dolce calore del sole tropicale...
Ora il sole picchia piu' forte, sei in pieno deserto e non c'e' traccia di oasi...
Ti svegli di soprassalto nella tua cabina di comandante del 'Neutronia'. Fa molto caldo. Troppo caldo. Ci dev'essere qualcosa che non funziona.

Linux:

```

#include "porting_universale.h"
using namespace std;

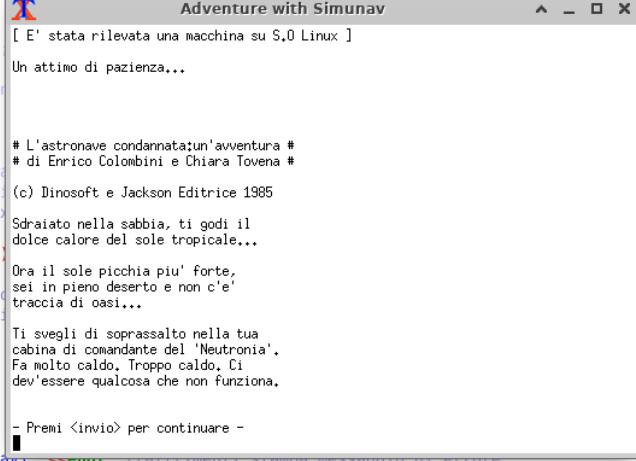
int main(int argc, char *argv[])
{
    #ifdef stoVenendoEseguitoSuMacchinaWindows
        system("title Adventure with Simunav")
        // system("mode con line=100");
        cout<<"[ E' stata rilevata una macchina su S.O Windows ]"
    #endif // defined

    #ifdef stoVenendoEseguitoSuMacchinaLinux
        cout<<"[ E' stata rilevata una macchina su S.O Linux ]"
        std::cout << "\033]0;" << "Adventure with Simunav"
    #endif // stoVenendoEseguitoSuMacchinaLinux

    fstream crono("Cronologia.odb",fstream::out);

    ifstream f("Mappa.nav",ios::in); //apre il file
    if (f.good()) //se è buono
    {
        Mappa M(f,crono); //istanzio la mappa
        Astro partita = Astro();
        partita.ciak(M,f);
    }
    else cout<<"Errore apertura file \"Mappa.nav\" - Premi <invio> per continuare -"
}

```



The terminal window shows the same text as the Windows version, indicating the program runs identically on both operating systems.

Il programma funziona quindi su entrambi i S.O. senza bisogno di ulteriori modifiche. A volte è possibile che, su Linux, la S appaia come un ‘.’ ma questo potrebbe essere un artefatto grafico dovuto alle impostazioni del pc in cui è stato testato e della macchina virtuale. Si richiedono test più approfonditi su vere macchine Linux.

ATTENZIONE: Su Linux è possibile che aprendo il file “porting_universale.cpp” con Codeblocks questo venga visualizzato senza testo:

Questo è solo un problema di visualizzazione in quanto il file contiene il testo. Basta premere con tasto

destro > proprietà per averne conferma:

2) APRIRE CORRETTAMENTE IL PROGETTO

Per aprire correttamente il progetto, bisogna assicurarsi di star aprendo, **con codeblocks**, il file “**bagordo.cbproj**” nella cartella Progetto Universale (o i vari test). Per evitare eventuali problemi, che si stia lavorando su Windows o su Linux, una volta caricato il progetto bisogna premere sul tasto “**Rebuild**” nella barra degli strumenti. Bisogna attendere che il compilatore ricostruisca il progetto e una volta finito si può lavorare normalmente, senza più necessità di fare ulteriori rebuild.

