**CORSO DI LAUREA IN INFORMATICA E   
COMUNICAZIONE DIGITALE**

Sede: Taranto

**ADVENTURE WITH SIMUNAV**Analisi, progettazione e realizzazione delle modifiche del progetto del gioco di avventura “L'astronave condannata”, presentato durante il corso di Algoritmi e Strutture Dati

**DOCENTE: Stefano Ferilli**

[1 Traccia 11](#_Toc23674302)

[1.1 Modifica Antonio Pastorelli 11](#_Toc23674303)

[1.2 Modifica Antonio Basile 11](#_Toc23674304)

[1.3 Modifica Del Giudice Angelo 11](#_Toc23674305)

[1.4 Modifica Moschetti Marco 11](#_Toc23674306)

[1.5 Modifica Germano Galeandro 11](#_Toc23674307)

[1.6 Modifica D'Andria Ivan – Dresda Claudio 12](#_Toc23674308)

[1.7 Modifica Federica Forte 12](#_Toc23674309)

[1.8 Modifica Gianluca Vacca 12](#_Toc23674310)

[1.9 Modifica Salvatore Vestita 12](#_Toc23674311)

[1.10 Modifica Vincenzo Giannuzzo 12](#_Toc23674312)

[1.11 Modifica Cicala Giacomo 12](#_Toc23674313)

[1.12 Modifica Scatigna Gianluca 12](#_Toc23674314)

[1.13 Modifica Della Folgore Grazia 12](#_Toc23674315)

[1.14 Modifica Palagiano Marcello 12](#_Toc23674316)

[1.15 Modifiche Chiarappa Rosa 13](#_Toc23674317)

[1.16 Modifiche Luceri Matteo 13](#_Toc23674318)

[2 Analisi 13](#_Toc23674319)

[2.1 Museo 13](#_Toc23674320)

[2.2 Scuola e Aule 13](#_Toc23674321)

[2.3 Archivio e Museo (vecchia versione del museo, inserita per completezza) 14](#_Toc23674322)

[2.4 Aula singola 14](#_Toc23674323)

[2.5 Ufficio Postale 14](#_Toc23674324)

[2.6 Dialoghi 15](#_Toc23674325)

[2.7 Sala scommesse 15](#_Toc23674326)

[2.8 Simulatore 15](#_Toc23674327)

[2.9 Auditorium 15](#_Toc23674328)

[2.10 Stazione 15](#_Toc23674329)

[2.11 Biblioteca 16](#_Toc23674330)

[2.12 Bari – Roma – Pisa 16](#_Toc23674331)

[2.13 Meccanico 16](#_Toc23674332)

[2.14 Verbo “ruba”, personaggio “carabiniere” e luogo “caserma” 17](#_Toc23674333)

[2.15 Zaino e Valigia 17](#_Toc23674334)

[3 Progettazione 18](#_Toc23674335)

[3.1 Museo 18](#_Toc23674336)

[3.2 Scuola e Aule 18](#_Toc23674337)

[3.2.1 Luogo Scuola 18](#_Toc23674338)

[3.2.2 Luogo Classe 1A 18](#_Toc23674339)

[3.2.3 Luogo Classe 2A 18](#_Toc23674340)

[3.2.4 Luogo Classe 3A 19](#_Toc23674341)

[3.3 Archivio e Museo (vecchia versione del museo) 19](#_Toc23674342)

[3.3.1 Luogo Archivio 19](#_Toc23674343)

[3.3.2 File 19](#_Toc23674344)

[3.3.3 Museo 19](#_Toc23674345)

[3.4 Aula singola 20](#_Toc23674346)

[3.4.1 Computer 20](#_Toc23674347)

[3.4.2 Sedie 20](#_Toc23674348)

[3.4.3 Proiettore 20](#_Toc23674349)

[3.4.4 Libri 20](#_Toc23674350)

[3.4.5 Lezione 20](#_Toc23674351)

[3.5 Ufficio Postale 21](#_Toc23674352)

[3.6 Dialoghi 21](#_Toc23674353)

[3.7 Sala scommesse e simulatore 22](#_Toc23674354)

[3.7.1 Azioni 22](#_Toc23674355)

[3.7.2 Inserimento Azione "Guarda simulatore" 22](#_Toc23674356)

[3.7.3 Inserimento Azione "Avvia simulatore" e "Gioca simulatore" 22](#_Toc23674357)

[3.8 Auditorium 23](#_Toc23674358)

[3.9 Stazione 23](#_Toc23674359)

[3.10 Palestra 24](#_Toc23674360)

[3.11 Sala giochi 24](#_Toc23674361)

[3.11.1 Slot machine 25](#_Toc23674362)

[3.12 Biblioteca 25](#_Toc23674363)

[3.13 Bari – Roma – Pisa 26](#_Toc23674364)

[3.14 Meccanico 26](#_Toc23674365)

[3.15 creazione verbo “ruba”, personaggio “carabiniere” e luogo “caserma” 27](#_Toc23674366)

[3.16 Zaino e valigia 27](#_Toc23674367)

[4 Realizzazione 29](#_Toc23674368)

[4.1 Museo 29](#_Toc23674369)

[4.2 Scuola e Aule 30](#_Toc23674370)

[4.2.1 Scuola e le classi 1A, 2A, 3A 30](#_Toc23674371)

[4.2.2 Oggetti 30](#_Toc23674372)

[4.2.3 Azioni 31](#_Toc23674373)

[4.3 Archivio e Museo (vecchia versione del museo) 31](#_Toc23674374)

[4.3.1 Archivio e Museo 32](#_Toc23674375)

[4.3.2 Oggetti 32](#_Toc23674376)

[4.3.3 Azioni 33](#_Toc23674377)

[4.4 Aula singola 33](#_Toc23674378)

[4.4.1 Aula 33](#_Toc23674379)

[4.4.2 Oggetti 33](#_Toc23674380)

[4.4.3 Azioni 34](#_Toc23674381)

[4.4.4 Comandi 34](#_Toc23674382)

[4.5 Ufficio Postale 35](#_Toc23674383)

[4.5.1 Mappa.nav 35](#_Toc23674384)

[4.5.2 24.txt 35](#_Toc23674385)

[4.5.3 Inserimento oggetto "Sportello Telematico" 35](#_Toc23674386)

[4.5.4 Inserimento oggetto "Documento d'Identità" 36](#_Toc23674387)

[4.5.5 Inserimento oggetto "Lettera" 37](#_Toc23674388)

[4.5.6 Inserimento oggetto "Pacco" 37](#_Toc23674389)

[4.6 Dialoghi 37](#_Toc23674390)

[4.7 Sala scommesse e simulatore 38](#_Toc23674391)

[4.7.1 Aggiunta oggetti e vocaboli nel gioco 38](#_Toc23674392)

[4.8 Auditorium 38](#_Toc23674393)

[4.8.1 Classi create 38](#_Toc23674394)

[4.8.2 Realizzazioni usate 39](#_Toc23674395)

[4.8.3 Modifica Mappa 39](#_Toc23674396)

[4.8.4 Oggetti 40](#_Toc23674397)

[4.8.5 Azioni 40](#_Toc23674398)

[4.9 Stazione 41](#_Toc23674399)

[4.10 Palestra 42](#_Toc23674400)

[4.10.1 Oggetti 42](#_Toc23674401)

[4.10.2 Azioni 43](#_Toc23674402)

[4.10.3 Comandi 43](#_Toc23674403)

[4.11 Sala Giochi 44](#_Toc23674404)

[4.11.1 Oggetto 44](#_Toc23674405)

[4.11.2 Azioni 45](#_Toc23674406)

[4.11.3 Comandi 45](#_Toc23674407)

[4.12 Biblioteca 45](#_Toc23674408)

[4.12.1 Oggetti 45](#_Toc23674409)

[4.12.2 Azioni 46](#_Toc23674410)

[4.12.3 Comandi 46](#_Toc23674411)

[4.13 Bari – Roma – Pisa 46](#_Toc23674412)

[4.13.1 Oggetti 47](#_Toc23674413)

[4.13.2 Azioni 47](#_Toc23674414)

[4.14 Modifica codici luoghi – Palagiano Marcello 48](#_Toc23674415)

[4.14.1 Modifica Mappa 48](#_Toc23674416)

[4.14.2 Modifica MappaOsservatorio 48](#_Toc23674417)

[4.14.3 Modifica MappaAliena 48](#_Toc23674418)

[4.15 Meccanico 49](#_Toc23674419)

[4.15.1 Oggetti 49](#_Toc23674420)

[4.15.2 Azioni 49](#_Toc23674421)

[4.16 CREAZIONE VERBO “RUBA”, PERSONAGGIO “CARABINIERE” E LUOGO “CASERMA” 50](#_Toc23674422)

[4.16.1 Verbi “ruba” e “rubati” 50](#_Toc23674423)

[4.16.2 Personaggio “carabiniere” 51](#_Toc23674424)

[4.16.3 Luogo “caserma” 51](#_Toc23674425)

[4.17 Zaino e valigia 52](#_Toc23674426)

[5 Implementazione 54](#_Toc23674427)

[5.1 Museo 54](#_Toc23674428)

[5.1.1 Museo.h 54](#_Toc23674429)

[5.1.2 Museo.cpp 55](#_Toc23674430)

[5.1.3 Lista.h 69](#_Toc23674431)

[5.1.4 Cella\_L\_DP.h 74](#_Toc23674432)

[5.1.5 Grafo.h 76](#_Toc23674433)

[5.1.6 Adiacenza.h 84](#_Toc23674434)

[5.1.7 Modifiche relative ad Astro.cpp 87](#_Toc23674435)

[5.2 Scuola e Aule 91](#_Toc23674436)

[5.2.1 Astro.h 91](#_Toc23674437)

[5.2.2 Gioco.h 91](#_Toc23674438)

[5.2.3 Astro.cpp 91](#_Toc23674439)

[5.2.4 Mappa.nav 109](#_Toc23674440)

[5.2.5 Aggiunta dei file nella cartella descrizioni 110](#_Toc23674441)

[5.3 Archivio e Museo (vecchia versione del museo) 111](#_Toc23674442)

[5.3.1 Astro.h 111](#_Toc23674443)

[5.3.2 Gioco.h 111](#_Toc23674444)

[5.3.3 Astro.cpp 111](#_Toc23674445)

[5.3.4 Gioco.cpp 125](#_Toc23674446)

[5.3.5 Mappa.nav 154](#_Toc23674447)

[5.3.6 Descrizioni 154](#_Toc23674448)

[5.4 Aula singola 155](#_Toc23674449)

[5.4.1 Aggiornamento dei vocaboli del dizionario 155](#_Toc23674450)

[5.4.2 Aggiornamento del dizionario degli oggetti 155](#_Toc23674451)

[5.4.3 Aggiornamento delle azioni di gioco 155](#_Toc23674452)

[5.4.4 Implementazione dell’Aula 166](#_Toc23674453)

[5.4.5 Correzione Funzionalità usa computer (Andrea Tursi) 166](#_Toc23674454)

[5.5 Ufficio Postale 168](#_Toc23674455)

[5.5.1 Inserimento del luogo "Ufficio Postale" 168](#_Toc23674456)

[5.5.2 Gioco.h 170](#_Toc23674457)

[5.5.3 Codice per inserimento oggetti 170](#_Toc23674458)

[5.5.4 Nuove azioni 171](#_Toc23674459)

[5.5.5 Modifiche al progetto di Bellanova 179](#_Toc23674460)

[5.6 Dialoghi 180](#_Toc23674461)

[5.6.1 Implementazione delle variabili 180](#_Toc23674462)

[5.6.2 Implementazione dei metodi 180](#_Toc23674463)

[5.6.3 Modifiche dei metodi già presenti in Gioco.cpp 184](#_Toc23674464)

[5.6.4 File aggiunti 194](#_Toc23674465)

[5.7 Sala scommesse e simulatore 209](#_Toc23674466)

[5.7.1 Implementazione luogo Sala scommesse e nuove azioni nel gioco 209](#_Toc23674467)

[5.7.2 Strutture dati aggiuntive 213](#_Toc23674468)

[5.8 Auditorium 218](#_Toc23674469)

[5.8.1 Gioco.h 218](#_Toc23674470)

[5.8.2 Astro.h 219](#_Toc23674471)

[5.8.3 Astro.cpp 219](#_Toc23674472)

[5.8.4 Gioco.cpp 228](#_Toc23674473)

[5.9 Stazione 229](#_Toc23674474)

[5.9.1 Mappa.nav 230](#_Toc23674475)

[5.9.2 Descrizioni (33.txt) 230](#_Toc23674476)

[5.9.3 Astro.cpp 230](#_Toc23674477)

[5.9.4 Astro.h 237](#_Toc23674478)

[5.9.5 Biglietto.h 237](#_Toc23674479)

[5.9.6 Biglietto.cpp 238](#_Toc23674480)

[5.10 Palestra 239](#_Toc23674481)

[5.10.1 Mappa.nav 239](#_Toc23674482)

[5.10.2 Descrizioni (34.txt) 239](#_Toc23674483)

[5.10.3 Astro.cpp 240](#_Toc23674484)

[5.10.4 Astro.h 242](#_Toc23674485)

[5.10.5 Gioco.cpp 243](#_Toc23674486)

[5.10.6 Gioco.h 244](#_Toc23674487)

[5.10.7 StatoFisico.h 244](#_Toc23674488)

[5.10.8 StatoFisico.cpp 245](#_Toc23674489)

[5.10.9 Palestra.h 246](#_Toc23674490)

[5.10.10 Scheda.h 246](#_Toc23674491)

[5.11 Sala giochi 247](#_Toc23674492)

[5.11.1 Aggiornamento mappa 247](#_Toc23674493)

[5.11.2 Aggiornamento dei vocaboli 248](#_Toc23674494)

[5.11.3 Aggiornamento azioni 248](#_Toc23674495)

[5.11.4 Aggiornamento oggetti 248](#_Toc23674496)

[5.11.5 Aziona gioca slot-machine 249](#_Toc23674497)

[5.11.6 Metodo load() e save() 252](#_Toc23674498)

[5.12 Implementazione Biblioteca 253](#_Toc23674499)

[5.12.1 Aggiornamento mappa 253](#_Toc23674500)

[5.12.2 Aggiornamento dei vocaboli 254](#_Toc23674501)

[5.12.3 Aggiornamento azioni 254](#_Toc23674502)

[5.12.4 Aggiornamento oggetti 254](#_Toc23674503)

[5.12.5 Dichiarazione azioni 254](#_Toc23674504)

[5.12.6 Implementazione metodi per funzionamento biblioteca 258](#_Toc23674505)

[5.12.7 ChangeLog 263](#_Toc23674506)

[5.13 Implementazione Bari – Roma – Pisa 268](#_Toc23674507)

[5.13.1 Mappa.nav 269](#_Toc23674508)

[5.13.2 Descrizioni(cartella) 269](#_Toc23674509)

[5.13.3 Astro.cpp 269](#_Toc23674510)

[5.13.4 Astro.h 277](#_Toc23674511)

[5.13.5 Gioco.cpp 277](#_Toc23674512)

[5.13.6 Gioco.h 279](#_Toc23674513)

[5.13.7 Luogo.cpp 279](#_Toc23674514)

[5.13.8 Luogo.h 279](#_Toc23674515)

[5.13.9 Oggetti.cpp 280](#_Toc23674516)

[5.13.10 Oggetti.h 280](#_Toc23674517)

[5.13.11 Oggetto.h 280](#_Toc23674518)

[5.13.12 Oggetto.cpp 280](#_Toc23674519)

[5.13.13 Veicolo.h 281](#_Toc23674520)

[5.13.14 Veicolo.cpp 282](#_Toc23674521)

[5.13.15 Bagagliaio.h 283](#_Toc23674522)

[5.13.16 Bagagliaio.cpp 284](#_Toc23674523)

[5.13.17 Autobus.h 286](#_Toc23674524)

[5.13.18 Autobus.cpp 286](#_Toc23674525)

[5.13.19 Automobile.h 287](#_Toc23674526)

[5.13.20 Automobile.cpp 287](#_Toc23674527)

[5.14 Modifica codici luoghi – Palagiano Marcello 288](#_Toc23674528)

[5.14.1 Modifica file “Mappa.nav” 288](#_Toc23674529)

[5.14.2 Modifica file “MappaOsservatorio.nav” 289](#_Toc23674530)

[5.14.3 Modifica file “MappaAliena.nav” 290](#_Toc23674531)

[5.14.4 Modifica file “Trasporti.nav” 291](#_Toc23674532)

[5.14.5 Modifica righe di codice 291](#_Toc23674533)

[5.15 Implementazione Meccanico 296](#_Toc23674534)

[5.15.1 Aggiunta del file “Batteria.h” 297](#_Toc23674535)

[5.15.2 Aggiunta del file “Batteria.cpp” 297](#_Toc23674536)

[5.15.3 Aggiunta del file “Attivita.h” 298](#_Toc23674537)

[5.15.4 Aggiunta del file “Attivita.cpp” 299](#_Toc23674538)

[5.15.5 Modifica al file “Gioco.cpp” 300](#_Toc23674539)

[5.15.6 Modifica al file “Gioco.h” 301](#_Toc23674540)

[5.15.7 Modifica al file “Astro.h” 302](#_Toc23674541)

[5.15.8 Modifica al file “Astro.cpp” 303](#_Toc23674542)

[5.15.9 Modifica al file “Mappa.nav” 315](#_Toc23674543)

[5.16 CREAZIONE VERBO “RUBA”, PERSONAGGIO ”CARABINIERE” E LUOGO “CASERMA” 316](#_Toc23674544)

[5.16.1 Verbo “ruba” 316](#_Toc23674545)

[5.16.2 Verbo “rubati” 316](#_Toc23674546)

[5.16.3 Personaggio”carabiniere” 316](#_Toc23674547)

[5.16.4 Luogo “caserma” 317](#_Toc23674548)

[5.17 Zaino e valigia 317](#_Toc23674549)

[5.17.1 Astro.h 317](#_Toc23674550)

[5.17.2 Astro.cpp 317](#_Toc23674551)

[5.17.3 Gioco.h 324](#_Toc23674552)

[5.17.4 Gioco.cpp 325](#_Toc23674553)

[5.17.5 Mappa.h 339](#_Toc23674556)

[5.17.6 Interfaccia.cpp 339](#_Toc23674557)

[5.17.7 Oggetti.h 340](#_Toc23674558)

[5.17.8 Oggetti.cpp 340](#_Toc23674559)

[5.17.9 Oggetto.h 342](#_Toc23674560)

[5.17.10 Oggetto.cpp 343](#_Toc23674561)

[5.17.11 Zaino.h 343](#_Toc23674562)

[5.17.12 Valigia.h 343](#_Toc23674563)

[6 Strutture intercambiabili 344](#_Toc23674564)

[*6.1* *Lista* 344](#_Toc23674565)

[6.1.1 Lista con Puntatore 344](#_Toc23674566)

[6.1.2 Lista con Vettore sequenziale 350](#_Toc23674567)

[6.1.3 Lista con puntatori doppi 353](#_Toc23674568)

[6.2 Lista Ordinata 357](#_Toc23674569)

[6.2.1 Lista Ordinata con vettore: 357](#_Toc23674570)

[6.2.2 Lista Ordinata con puntatori: 359](#_Toc23674571)

[6.3 Pila 360](#_Toc23674572)

[6.3.1 Pila con puntatori: 360](#_Toc23674573)

[6.4 Coda 362](#_Toc23674574)

[6.4.1 Coda con lista 363](#_Toc23674575)

[6.4.2 Coda con puntatori 365](#_Toc23674576)

[6.4.3 Coda con Vettore sequenziale 369](#_Toc23674577)

[6.5 *Coda di priorità* 370](#_Toc23674578)

[6.5.1 Coda con priorità con vettore (Heap) 370](#_Toc23674579)

[6.5.2 Coda con priorità con lista ordinata 376](#_Toc23674580)

[6.6 Albero n-ario 385](#_Toc23674581)

[6.6.1 AlberoNario: realizzazione con alberi binari 385](#_Toc23674582)

[6.6.2 AlberoNario: realizzazione primofiglio/fratello 395](#_Toc23674583)

[6.7 *Insiemi* 399](#_Toc23674584)

[6.7.1 Insiemi con lista 399](#_Toc23674585)

[6.8 Dizionario 403](#_Toc23674586)

[6.8.1 Dizionario con vettore ordinato 403](#_Toc23674587)

[6.8.2 Dizionario con lista 406](#_Toc23674589)

[6.9 Grafi 408](#_Toc23674590)

[6.9.1 Matrice di Adiacenza 408](#_Toc23674591)

[6.9.2 Vettore di Liste di Adiacenza 416](#_Toc23674592)

# 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.

## Modifica Antonio Pastorelli

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

## Modifica Antonio Basile

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

## 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.

## 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.

## 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.

## Modifica D'Andria Ivan – Dresda Claudio

Aggiungere la funzionalità di dialogo tra alcuni personaggi non giocanti presenti nell'adventure 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.

## 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").

## Modifica Gianluca Vacca

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

## 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").

## 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”).

## 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”.

## 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.

## 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.

## 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).

## 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”

## Modifiche Luceri Matteo

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

# 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).

## 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.

## 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.

## 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 cosi 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.

## Aula singola

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

## Ufficio Postale

Nell'Ufficio postale sarà possibile inviare e ricevere oggetti e si potrà interagire con la Banca.  
L'Ufficio Postale esplicherà le sue funzioni tramite uno Sportello telematico che richiederà un documento d'identità per il servizio di ricezione e darà accesso a due nuovi oggetti : Una Lettera con dei suggerimenti e un Pacco con un oggetto misterioso (la chiave del secondo pilota).  
Per accedere allo Sportello telematico si dovrà prima attendere il proprio turno in base a una fila randomica di persone. Sarà comunque possibile scegliere se aspettare il proprio turno o ritentare più in la.  
Tutte le funzionalità dell'Ufficio Postale non sono comunque necessarie ai fini del completamento del gioco.

## Dialoghi

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

## Sala scommesse

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

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

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

## Simulatore

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

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

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

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

## 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.

## Stazione

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

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

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

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

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

## 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.

## Bari – Roma – Pisa

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

* Autobus
* Automobile
* Chiave automobile
* Ticket Bus

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

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

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

## Meccanico

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

## Verbo “ruba”, personaggio “carabiniere” e luogo “caserma”

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

## Zaino e Valigia

Le funzionalità da implementare nel progetto di base sono:

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

Lo zaino e la valigia hanno dei limiti di peso trasportabile. La posizione di questi due contenitori sarà fissa: ad ogni nuova partita si troveranno sempre nello stesso luogo, con la possibilità di scegliere se utilizzarli o meno. Non sarà possibile trasportare entrambi gli oggetti contenitori, bisognerà scegliere solo uno dei due.  
  
Nella valigia si possono inserire e prelevare quanti oggetti si vuole, rimanendo nei limiti del peso massimo consentito, invece nello zaino si potranno sì mettere quanti oggetti si vuole rimanendo nei limiti del peso, ma si potrà prendere solo il primo di questi.  
Ciò comporta che, nel caso in cui si voglia esplorare tutto lo zaino, bisognerà togliere tutti gli oggetti presenti. L’utilizzo dello zaino o della valigia è a discrezione dell’avventuriero che, in caso ce l’abbia, può decidere se trasportare con sé gli oggetti o metterli nel contenitore.

# Progettazione

## 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.

## Scuola e Aule

### Luogo Scuola

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

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

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

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

### Luogo Classe 1A

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

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

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

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

### Luogo Classe 2A

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

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

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

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

### 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.*

## Archivio e Museo (vecchia versione del museo)

### 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.

### 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.

### 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.

## 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.

### 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.

### 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.

### 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.

### 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.

### 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 unita di vita (unità di tempo) sia che la lezione sia utile, sia che essa sia inutile.

## 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.

## Dialoghi

1. Incontro personaggio
   1. Caricamento stato domanda corrente:
      * Se incontro il personaggio per la prima volta stampa presentazione e prima domanda
      * Se ho già iniziato un dialogo con il personaggio stampo l'ultima domanda con le proprie possibilità di risposte che erano state proposte precedentemente
2. Dialogo
   1. fino a quando non si decide di interrompere il dialogo o il personaggio termina le domande da porre:
      1. Stampo domanda e possibilità di risposte allo stato corrente
      2. Leggo risposta
      3. Aggiorno stato dialogo
3. Fine dialogo
   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.

## 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 vincita 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 |

### 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.

### 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.

### 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.

## 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).

## 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.

## 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.

## 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.

### 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.

## 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.

## 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.

## 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.

## creazione verbo “ruba”, personaggio “carabiniere” e luogo “caserma”

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

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

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

## Zaino e valigia

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

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

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

**“Se vuoi portare l’oggetto con te, premi (y);**

**se vuoi metterlo in valigia, premi (v)”**

in caso si sia raccolta la valigia oppure:

**“Se vuoi portare l’oggetto con te, premi (y);**

**se vuoi metterlo nello zaino, premi (z)”**

in caso si sia raccolto lo zaino.

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

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

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

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

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

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

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

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

In informatica ci si riferisce al modo estensionale.

La relazione fondamentale è quella di appartenenza di un elemento x ad un insieme i (x∈i), in base alla quale e poi definita la relazione di inclusione di un insieme i’ in un insieme i’’ (i^'⊆i^'' ).

Le operazioni principali fra due insiemi, i^' e i'' sono unione (i^'∪ i^'' ),intersezione (i^'∩ i^'' ) e differenza (i^' \ i^'' ), ben note in matematica. Un insieme che non ha elementi è detto vuoto e viene indicato con Ø.

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

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

È una struttura lineare e dinamica.

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

# Realizzazione

## 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.

## 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

### 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 = Sali, 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.

### 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 |

### 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 |

## 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 cbiblioreare conflitti con gli oggetti e i luoghi già presenti.

### 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.

### 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 |

### 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 |

## 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.

### 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).

### 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 |

### 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.

### 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 |

## Ufficio Postale

Per l'integrazione nell'adventure sarà in oltre 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.

### 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

### 24.txt

Nell'Ufficio Postale

Nell'Ufficio Postale

Di nuovo nell'Ufficio Postale

Nuovamente nell'Ufficio Postale

Ancora una volta nell'Ufficio Postale

### 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 combando "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 riceve 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 veranno 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)

### 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

### 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!"

### 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.

## 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)

## Sala scommesse e simulatore

### 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.

#### Vocaboli

vocabolario.inserisci("simulatore",58);

vocabolario.inserisci("avvia",14);

vocabolario.inserisci("gioca",44);

#### 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

#### Oggetti

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

## 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

### 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)

### 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 punti 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.

### Modifica Mappa

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

L'auditorium ha:

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

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

* **Vecchia etichetta:** 0204120619000
* **Nuova etichetta:** 0204120619332

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

### 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 giocola 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 prendibile 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 |

### 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 |

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

Ad ogni oggetto sono associate le seguenti caratteristiche:

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

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

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

|  |  |
| --- | --- |
| VOCABOLO | CODICE OGGETTO |
| Sul | 7 |
| Sulla | 7 |

Ad ogni azione sono state assegnate le seguenti caratteristiche univoche:

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

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

Per ogni comando sono state assegnate le seguenti caratteristiche univoche:

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

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

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

|  |  |  |
| --- | --- | --- |
| CODICE DEL LUOGO | NOME DEL LUOGO | CODICE DIREZIONI |
| 34 | Palestra | 000000000500 |

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

### 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 giocato 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 |

### 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 |

### 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 |

## 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, cioe “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 indentifica 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.

### Oggetto

* **Panca**: l’oggetto chiamato “Slot-Machine” è un oggetto non trasportabile. Il giocatore potrà usare la slot-machine e decidere quante volte giocare finchè 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 |

### 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 |

### Comandi

Per ogni comando sono state assegnate le seguenti caratteristiche univoche:

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

|  |  |  |
| --- | --- | --- |
| CODICE AZIONE | CODIFICA | COMANDO |
| 125 | 3600290086 | usa slot-machine |

## 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 | 000000000028 |
| 38 | Vetrina | 000000000028 |

### 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 |

### 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 |

### 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 |

## Bari – Roma – Pisa

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

Arrivati alla “stazione di servizio”:

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

I luoghi sono definiti come segue:

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

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

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

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

### 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) |

### 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.

## 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.

### 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 |

### 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 |

### 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 |

## 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:

**NNSSEEOOUUDD**

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 |

### 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 | pila | 70 | -41 |
| delle batterie scariche | 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 |
|  |  |  |  |

### 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 |

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

### Verbi “ruba” e “rubati”

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

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

|  |  |
| --- | --- |
| **ETICHETTA** | **CODICE** |
| ruba | 50 |
| rubati | 200 |

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

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

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

* LL = Codice del luogo in cui il comando può essere impartito.
* VV = Codice del verbo che indica una certa azione.
* OO = Codice dell’oggetto.

**- Codice Azione:** codice univoco che rappresenta l’azione.

|  |  |  |
| --- | --- | --- |
| **VERBO** | **CODIFICA** | **CODICE** |
| ruba | 100509999 | -10 |
| rubati | 2000000 | 11 |

### 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

### Luogo “caserma”

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

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

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

|  |  |  |
| --- | --- | --- |
| **ETICHETTA** | **CODICE MAPPA** | **CODICE LUOGO** |
| Caserma | 030000000000 | 53 |

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

## Zaino e valigia

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

• Inserisci(“aiuto”, 259, vocabolario)

• Inserisci(“help”, 259, vocabolario)

• Inserisci(“valigia”, 155, vocabolario)

• Inserisci(“zaino”, 156, vocabolario)

• Inserisci(“raccogli”, 258, vocabolario)

E verranno inseriti i seguenti oggetti:

• Inserisci(Oggetto("una valigia", 155, 5, 0), oggetti)

• Inserisci(Oggetto("uno zaino", 156, 6, 0), oggetti)

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

• L’oggetto Manuale avrà peso 1;

• L’oggetto Tuta avrà peso 2;

• L’oggetto Camice avrà peso 3;

• L’oggetto Casco avrà peso 3;

Tutti i pesi sono espressi in kg.

Verranno aggiunte le seguenti azioni:

• Inserisci(80155, -2, azioni) //azione prendi valigia

• Inserisci(80156, -2, azioni) //azione prendi zaino

• Inserisci(90155, 3, azioni) //azione lascia valigia

• Inserisci(90156, 3, azioni) //azione lascia zaino

• Inserisci(100155, 140, azioni) //azione guarda valigia

• Inserisci(100156, 141, azioni) //azione guarda zaino

• Inserisci(1550000, 143, azioni) //azione inventario valigia

• Inserisci(1560000, 144, azioni) //azione inventario zaino

• Inserisci(2590000, 145, azioni) //azione aiuto/help

E verranno modificate le seguenti azioni:

• Inserisci(2050, -71, azioni)

• Inserisci(2051, -71, azioni)

• Inserisci(2052, -71, azioni)

in:

• Inserisci (200050, -142)

• Inserisci (200051, -142)

• Inserisci (200052, -142)

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

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

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

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

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

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

Mentre sono stati aggiunti i seguenti file:

* Zaino.h
* Valigia.h

# Implementazione

## Museo

In seguito il codice relativo alle strutture e classi implementate:

### 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

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

titoli[3]="Coproliti 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 e' fossilizzato. Dall'analisi di questi reperti, si possono ricavare informazioni sulle abitudini alimentari e l'habitat nel quale l'animale viveva. Per esempio, la presenza di semi, foglie, corteccia o radici indica che l'escremento e' stato prodotto da un erbivoro, mentre se si individuano frammenti di ossa, artigli o tendini l'animale era un carnivoro.";

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

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

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

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

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

descrizioni[6]="Come testimonianza di questo periodo storico potrete ammirare antichi affreschi. Per ulteriri 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 scorcio 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 â€œfalsa portaâ€�, mentre siede in compagnia della madre Henutes davanti a una tavola ricolma di vasellame da mensa e di alimenti (pani, anatre giÃ  spennate, tranci di carne bovina, etc.) utili per la sua sopravvivenza ultraterrena che lâ€™iscrizione in caratteri geroglifici sottostante completa per forza magica. Lâ€™importanza del pane, quale alimento base della dieta egiziana, appare evidente, anche se non e' possibile differenziare le varie tipologie di pane elencate per qualitÃ  di ingredienti, modalitÃ  di impasto e di cottura.";

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

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

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

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

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

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

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

descrizioni[17]="La decorazione principale di questo sarcofago destinato al corredo funerario di un egiziano di nome Irinimenpu consiste in una serie di pannelli a 'facciata di palazzo', un motivo ripreso dall'architettura funararia dell'Antico Regno, che rende il sarcofago simile a una dimora eterna. Su uno dei lati lunghi della cassa, in posizione centrale, sono raffigurate numerose offerte alimentari (pani, ortaggi, frutti, tranci di carni bovine, volatili giÃ  spennati, contenitori per liquidi, etc.), perche' il defunto possa nutrirsene per forza magica nella vita ultraterrena. Sullo stesso lato,a una delle estremita', e' dipinta una porta chiusa, sopra la quale sono collocati due occhi che indicano la presenza all'interno della testa della mummia e allo stesso tempo rappresentano una protezione magica per il corpo del defunto. L'ottimo stato di conservazione del legno e dei vivaci colori utilizzati per decorarlo si deve al clima caldo e asciutto dell'Egitto,oltre che alla collocazione originaria del sarcofago nell'ambiente protetto della 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 e' l'area geografica della penisola italiana meridionale che fu anticamente colonizzata dai Greci a partire dall' VIII secolo a.C.. Di questa civilta' 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 Tara alla fine del IV secolo AC.";

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

descrizioni[20]="Raffigurato nell'atto di offrire una melagrana ad un serpente, simbolo funerario delle divinita'  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]="E' raffigurata l'immagine di una Musa, o di Afrodite, circondata da Eroti. 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 piu' significativi dell'arte greca, e tra le testimonianze dirette dei grandi maestri scultori dell'eta' 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 cittÃ  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);

}

### Lista.h

#ifndef LISTA\_H\_

#define LISTA\_H\_

#include "Cella\_L\_DP.h"

template <class tipoelem>

class Lista{

public:

typedef Cella\_L\_DP<tipoelem>\* posizione;

Lista();

Lista(const Lista&);

~Lista();

void crealista();

bool listavuota() const;

posizione primolista() const;

posizione succlista(posizione) const;

posizione preclista(posizione) const;

bool finelista(posizione) const;

tipoelem leggilista(posizione) const;

void inslista(tipoelem, posizione&);

void scrivilista(tipoelem, posizione);

void canclista(posizione&);

private:

posizione lista;

};

template <class tipoelem>

Lista<tipoelem>::Lista()

{

crealista();

}

template <class tipoelem>

Lista<tipoelem>::Lista(const Lista& b)

{

crealista();

typename Lista<tipoelem>::posizione ind = primolista(),ind2 = b.primolista();

while (!b.finelista(ind2))

{

inslista(b.leggilista(ind2),ind);

ind = succlista(ind);

ind2 = b.succlista(ind2);

}

}

template <class tipoelem>

Lista<tipoelem>::~Lista()

{

posizione p = primolista();

posizione temp;

while(p->getSucc() != nullptr)

{

temp = p;

p=p->getSucc();

delete temp;

}

}

template <class tipoelem>

void Lista<tipoelem>::crealista()

{

lista = new Cella\_L\_DP<tipoelem>;

lista->setSucc(nullptr);

lista->setPrec(nullptr);

}

template <class tipoelem>

bool Lista<tipoelem>::listavuota() const

{

return (lista->getSucc() == nullptr && lista->getPrec() == nullptr);

}

template <class tipoelem>

typename Lista<tipoelem>::posizione Lista<tipoelem>::primolista() const

{

return lista;

}

template <class tipoelem>

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

{

if(lista->getSucc() == nullptr)

return p;

else

return p->getSucc();

}

template <class tipoelem>

typename Lista<tipoelem>::posizione Lista<tipoelem>::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>::leggilista(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());

succlista(p)->setPrec(preclista(p));

}

p = p->getSucc();

delete temp;

}

#endif //LISTA\_H\_

### Cella\_L\_DP.h

#ifndef CELLA\_L\_DP\_H\_

#define CELLA\_L\_DP\_H\_

template <class tipoelem>

class Cella\_L\_DP{

public:

Cella\_L\_DP();

~Cella\_L\_DP();

tipoelem getElem() const;

void setElem(tipoelem);

Cella\_L\_DP<tipoelem>\* getSucc() const;

void setSucc(Cella\_L\_DP<tipoelem>\*);

Cella\_L\_DP<tipoelem>\* getPrec() const;

void setPrec(Cella\_L\_DP<tipoelem>\*);

private:

tipoelem elem;

Cella\_L\_DP<tipoelem>\* succ;

Cella\_L\_DP<tipoelem>\* prec;

};

template <class tipoelem>

Cella\_L\_DP<tipoelem>::Cella\_L\_DP()

{

succ = nullptr;

prec = nullptr;

}

template <class tipoelem>

Cella\_L\_DP<tipoelem>::~Cella\_L\_DP(){}

template <class tipoelem>

tipoelem Cella\_L\_DP<tipoelem>::getElem() const

{

return elem;

}

template <class tipoelem>

void Cella\_L\_DP<tipoelem>::setElem(tipoelem e)

{

elem = e;

}

template <class tipoelem>

Cella\_L\_DP<tipoelem>\* Cella\_L\_DP<tipoelem>::getSucc() const

{

return succ;

}

template <class tipoelem>

void Cella\_L\_DP<tipoelem>::setSucc(Cella\_L\_DP<tipoelem>\* e)

{

succ = e;

}

template <class tipoelem>

Cella\_L\_DP<tipoelem>\* Cella\_L\_DP<tipoelem>::getPrec() const

{

return prec;

}

template <class tipoelem>

void Cella\_L\_DP<tipoelem>::setPrec(Cella\_L\_DP<tipoelem>\* e)

{

prec = e;

}

#endif //CELLA\_L\_DP\_H\_

### 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 grafovuoto() 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>::grafovuoto() 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 restituitre (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.finelista(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 appartenenente 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.finelista(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 appartenenente 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.finelista(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

### 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);

}

//sovraccarico output

template<class nodo,class tipoPeso> ostream& operator<<(ostream& os, const Adiacenza<nodo,tipoPeso>& a)

{

os<<"("<<a.legginodo()<<"|"<<a.leggipeso()<<")";

return(os);

}

#endif

### Modifiche relative ad Astro.cpp

void Astro::azione\_96(){

//system("cls"); windows

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

int num\_persone=(rand()+1)%11;

bool uscita\_museo=false;

bool uscita\_coda=false;

string risposta\_attesa; //input per la risposta

int risposta\_documento; //input per la risposta al documento

while (!uscita\_coda)

{

if (num\_persone==0)

{

uscita\_coda=true;

}

else

{

interfaccia.scrivi\_parziale("Ci sono ");

interfaccia.scrivi\_parziale(num\_persone);

interfaccia.scrivi(" persone in coda per pagare il biglietto /n");

interfaccia.scrivi("Il costo del biglietto e' 10 euro");

interfaccia.scrivi("Desideri aspettare e pagare? (s/n):");

cin >> risposta\_attesa;

if (risposta\_attesa=="s" || risposta\_attesa=="sÃ¬" || risposta\_attesa=="si")

{ bool controlla=false;

tempo=tempo-num\_persone;

uscita\_coda=true;

controlla=biglietto\_museo();

if(controlla==false)

uscita\_museo=true;

}

else

if (risposta\_attesa=="n" || risposta\_attesa=="no")

{

uscita\_museo=true;

uscita\_coda=true;

}

else

{

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

}

}

}

//INIZIO MODIFICHE ANTONIO PASTORELLI

Museo museo;

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

while (!uscita\_museo)

{

copiaLista(listastanze,museo.getstanzeadiacenti());

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

cout<<endl<<museo.leggiStanza(museo.getstanzacorrente()).gettitolo()<<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.leggilista(indice)).gettitolo()<<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);

}

## Scuola e Aule

### 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

### 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;

### Astro.cpp

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

I vocaboli inseriti sono:

vocabolario.inserisci("scuola", 16);

vocabolario.inserisci("1A", 16);

vocabolario.inserisci("2A", 16);

vocabolario.inserisci("3A", 16);

vocabolario.inserisci("cartina", 60);

vocabolario.inserisci("banchi", 43);

vocabolario.inserisci("sedia", 56);

vocabolario.inserisci("registro", 88);

vocabolario.inserisci("lavagna", 90);

vocabolario.inserisci("cattedra", 70);

//inserimento di personaggi

vocabolario.inserisci("preside", 73);

vocabolario.inserisci("maestra", 73);

Gli oggetti inseriti sono:

//Oggetti inseriti nella scuola

oggetti.inserisci(Oggetto("una cartina Galattica", 60,-28));

oggetti.inserisci(Oggetto("il Preside", 73, -28));

//1A

oggetti.inserisci(Oggetto("la maestra Clara", 73, -29));

oggetti.inserisci(Oggetto("una cartina geografica", 60, -29));

oggetti.inserisci(Oggetto("dei banchi", 43, -29));

oggetti.inserisci(Oggetto("delle sedie", 56, -29));

oggetti.inserisci(Oggetto("una cattedra", 70, -29));

//2A

oggetti.inserisci(Oggetto("la maestra Mara", 73, -30));

oggetti.inserisci(Oggetto("dei banchi", 43, -30));

oggetti.inserisci(Oggetto("delle sedie", 56, -30));

oggetti.inserisci(Oggetto("un registro", 88, 30));

oggetti.inserisci(Oggetto("una cattedra", 70, -30));

//3A

oggetti.inserisci(Oggetto("dei banchi", 43, -31));

oggetti.inserisci(Oggetto("delle sedie", 56, -31));

oggetti.inserisci(Oggetto("una cattedra", 70, -31));

oggetti.inserisci(Oggetto("una lavagna", 90, -31));

Le azioni inserite sono:

azioni.inserisci(2588,97); //leggi registro

azioni.inserisci(1088, 97); //guarda registro

azioni.inserisci(311090, 98); //guarda lavagna

azioni.inserisci(283973, 100); //parla con il preside

azioni.inserisci(293973, 101); //parla con la maestra Clara

azioni.inserisci(303973, 102); //parla con la maestra Mara

azioni.inserisci(281060, 103); //guarda cartina galattica

azioni.inserisci(291060, 104); //guarda cartina geografica

Sono state inizializzate le variabili booleane a false:

gioco\_impiccato = false;

gioco\_preside = false;

gioco\_prof\_clara = false;

gioco\_prof\_mara = false;

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

case 97:

azione\_97();

break;

case 98:

azione\_98();

break;

case 100:

azione\_100();

break;

case 101:

azione\_101();

break;

case 102:

azione\_102();

break;

case 103:

azione\_103();

break;

case 104:

azione\_104();

break;

Nello stesso file sono implementati le procedure void azione\_97() (guarda lavagna), void azione\_98() (gioco dell’impiccato), void azione\_100() (parla con il preside), void azione\_101() (parla con la maestra Clara), void azione\_102() (parla con la maestra Mara), void azione\_103() (guarda cartina Galattica), void azione\_104() (guarda cartina geografica).

//guarda registro

void Astro::azione\_97()

{

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

interfaccia.scrivi("|-------------- Registro degli Alunni 2A -----------------------|");

interfaccia.scrivi("|- Data: 16/02/2017.............................................|");

interfaccia.scrivi("|- Albano Giuseppe.......................................ASSENTE|");

interfaccia.scrivi("|- Albino Gianni.........................................ASSENTE|");

interfaccia.scrivi("|- Franco Walter........................................PRESENTE|");

interfaccia.scrivi("|- D'angelo Giuseppe.....................................ASSENTE|");

interfaccia.scrivi("|- Zagaria Marcello.....................................PRESENTE|");

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

}

//gioco dell'impiccato

void Astro::azione\_98()

{

system("cls");

string risp;

if(!gioco\_impiccato)

{

do

{

interfaccia.scrivi("c'e' scritto qualcosa.....");

interfaccia.scrivi("sembra che sia un gioco...");

interfaccia.scrivi("e' il gioco dell'impiccato!!!");

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

}

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

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

{

gioco\_impiccato = true;

string parola = "giocattolo";

string parola\_nascosta = "g\*\*\*\*t\*\*\*\*";

string parola\_indovinata;

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\_indovinata = interfaccia.leggi\_stringa("Scrivi parola: ");

if(parola == parola\_indovinata)

{

interfaccia.scrivi("La parola e': giocattolo");

interfaccia.scrivi("Hai Vinto!!!");

interfaccia.scrivi("Ti verra' incrementato il tempo di 40 punti!");

tempo = tempo + 40;

}

else

{

interfaccia.scrivi("La parola da indovinare era: giocattolo");

interfaccia.scrivi("Hai Perso!!!");

interfaccia.scrivi("Hai perso tempo ti verra' decrementato il tempo di 40 punti!");

tempo = tempo - 40;

}

}

else

{

trovato = false;

carattere = interfaccia.leggi\_carattere("Inserisci carattere: ");

for(int i = 1; i <= lunghezza; i++)

{

if(parola\_nascosta[i] == '\*')

{

if(parola[i] == carattere)

{

parola\_nascosta[i] = carattere;

trovato = true;

}

}

}

if(!trovato)

{

interfaccia.scrivi("Non hai indovinato nessun carattere");

num\_tentativi--;

}

else

{

controllo = false;

for(int i = 0; i <= lunghezza && !controllo; i++)

{

if(parola\_nascosta[i] == '\*')

controllo = true;

}

if(!controllo)

{

interfaccia.scrivi("La parola e': giocattolo");

interfaccia.scrivi("Hai Vinto!!!");

interfaccia.scrivi("Ti verra' incrementato il tempo di 40 punti!");

tempo = tempo + 40;

indovina = true;

}

}

}

}

while(num\_tentativi > 0 && !indovina);

if(num\_tentativi == 0)

{

interfaccia.scrivi("La parola da indovinare era: giocattolo");

interfaccia.scrivi("Hai Perso!!!");

interfaccia.scrivi("Hai perso tempo ti verra' decrementato il tempo di 40 punti!");

tempo = tempo - 40;

}

}

else

interfaccia.scrivi("Hai cose piu importanti da fare... salva l'astronave Neutronia!!!");

}

else

interfaccia.scrivi("Hai gia' giocato... salva l'astronave Neutronia!!!");

}

//parla con il preside

void Astro::azione\_100()

{

string nome;

string accetta;

string risposta;

if(!gioco\_preside)

{

interfaccia.scrivi("Salve, benvenuto nella Scuola Elementare della Neutronia");

nome = interfaccia.leggi\_stringa("Con chi ho il piacere di parlare?");

interfaccia.scrivi\_parziale("Quindi lei e' ");

interfaccia.scrivi\_parziale(nome);

interfaccia.a\_capo();

do

{

accetta = interfaccia.leggi\_stringa("Ho un problema da risolvere... Mi puo' aiutare??? [Si/No]");

}

while(accetta != "si" && accetta != "s" && accetta != "Si" &&

accetta != "no" && accetta != "n" && accetta != "No");

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

{

gioco\_preside = true;

interfaccia.scrivi("Benissimo!\n");

interfaccia.scrivi("Sto cercando di risolvere un cruciverba...");

risposta = interfaccia.leggi\_stringa("9 orizzontale.... un arma simile ad un arco...");

if(risposta == "balestra")

{

interfaccia.scrivi("Grazie mille!!!");

interfaccia.scrivi("Il tuo tempo e' stato incrementato di 10 punti!!!");

tempo = tempo + 10;

}

else

{

interfaccia.scrivi("Hai Sbagliato!!!");

interfaccia.scrivi("La parola era BALESTRA");

interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti!!!");

tempo = tempo - 10;

}

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

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

if(risposta == "goccia")

{

interfaccia.scrivi("Grazie mille!!!");

interfaccia.scrivi("Il tuo tempo e' stato incrementato di 10 punti!!!");

tempo = tempo + 10;

}

else

{

interfaccia.scrivi("Hai Sbagliato!!!");

interfaccia.scrivi("La parola era Goccia");

interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti!!!");

tempo = tempo - 10;

}

interfaccia.scrivi("Un'altra domanda...");

risposta = interfaccia.leggi\_stringa("10 verticale.... Interruttore per... liquidi: ");

if(risposta == "rubinetto")

{

interfaccia.scrivi("Grazie mille!!!");

interfaccia.scrivi("Il tuo tempo e' stato incrementato di 10 punti!!!");

tempo = tempo + 10;

}

else

{

interfaccia.scrivi("Hai Sbagliato!!!");

interfaccia.scrivi("La parola era Rubinetto");

interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti!!!");

tempo = tempo - 10;

}

}

else

{

interfaccia.scrivi\_parziale("Ciao ");

interfaccia.scrivi\_parziale(nome);

interfaccia.a\_capo();

}

}

else

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

}

//parla con la maestra Clara

void Astro::azione\_101()

{

if(!gioco\_prof\_clara)

{

string risp;

do

{

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

risp = interfaccia.leggi\_stringa("Insegno storia, vuoi fare un test? [Si/No]");

}

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

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

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

{

string risposta;

gioco\_prof\_clara = true;

interfaccia.scrivi("Rispondi alle seguenti domande:");

risposta = interfaccia.leggi\_stringa("Prima domanda: Quando cadde l'Impero Romano d'Occidente?");

if(risposta == "476 d.C.")

{

tempo = tempo + 5;

interfaccia.scrivi("Risposta Esatta");

interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti");

}

else

{

tempo = tempo - 10;

interfaccia.scrivi("Risposta sbagliata");

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

}

interfaccia.a\_capo();

risposta = interfaccia.leggi\_stringa("Seconda domanda: Quando venne scoperta l'America?");

if(risposta == "1492")

{

tempo = tempo + 5;

interfaccia.scrivi("Risposta Esatta");

interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti");

}

else

{

tempo = tempo - 10;

interfaccia.scrivi("Risposta sbagliata");

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

}

interfaccia.a\_capo();

risposta = interfaccia.leggi\_stringa("Terza Domanda: Quando esplose la prima Guerra Mondiale?");

if(risposta == "1914")

{

tempo = tempo + 5;

interfaccia.scrivi("Risposta Esatta");

interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti");

}

else

{

tempo = tempo - 10;

interfaccia.scrivi("Risposta sbagliata");

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

}

}

else

interfaccia.scrivi("Ok ci vediamo!");

}

else

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

}

//parla con la maestra mara

void Astro::azione\_102()

{

if(!gioco\_prof\_mara)

{

string risp;

do

{

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

risp = interfaccia.leggi\_stringa("Insegno matematica, Vuoi fare un test? [Si/No]");

}

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

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

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

{

gioco\_prof\_mara = true;

int num\_esatte;

string risp1, risp2, risp3;

interfaccia.scrivi("Risolvi le seguenti espressioni: ");

interfaccia.scrivi("1. [(3\*4)/2+5\*6-(10/5)\*6]");

interfaccia.scrivi("2. [6\*7-18\*(5-12)]");

interfaccia.scrivi("3. [(4^2/8)\*7+(2^3)\*4]");

interfaccia.a\_capo();

risp1 = interfaccia.leggi\_stringa("Risultato della prima espressione: ");

if(risp1 == "24")

{

interfaccia.scrivi("Risposta Corretta! Bravo!!!");

interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti.");

tempo = tempo + 5;

}

else

{

interfaccia.scrivi("Risposta Sbagliata!");

interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti");

tempo = tempo - 10;

}

risp2 = interfaccia.leggi\_stringa("Risultato della seconda espressione: ");

if(risp2 == "168")

{

interfaccia.scrivi("Risposta Corretta! Bravo!!!");

interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti.");

tempo = tempo + 5;

}

else

{

interfaccia.scrivi("Risposta Sbagliata!");

interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti");

tempo = tempo - 10;

}

risp3 = interfaccia.leggi\_stringa("Risultato della terza esprezzione: ");

if(risp3 == "46")

{

interfaccia.scrivi("Risposta Corretta! Bravo!!!");

interfaccia.scrivi("il tuo tempo e' stato incrementato di 5 punti.");

tempo = tempo + 5;

}

else

{

interfaccia.scrivi("Risposta Sbagliata!");

interfaccia.scrivi("Il tuo tempo e' stato decrementato di 10 punti");

tempo = tempo - 10;

}

}

else

{

interfaccia.scrivi("Ok ci vediamo");

}

}

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");

}

### 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

### 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

## Archivio e Museo (vecchia versione del museo)

### 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

### Gioco.h

Aggiunti 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

### 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 dallo 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");

}

}

}

### Gioco.cpp

All’interno della classe Gioco.cpp sono state realizzate delle funzioni e delle procedure di supporto all’azione 96 inerenti al luogo “Museo”:

* biglietto\_museo
* mostra\_galleria\_a
* mostra\_galleria\_b
* mostra\_galleria\_c
* mostra\_galleria\_d

La funzione *biglietto\_museo* è di tipo booleano, la sua funzione è quella di controllare se il giocatore sia in possesso del portafoglio o del credito disponibile per l’acquisto del biglietto.   
In caso di esito negativo la funzione restituisce “false” e stampa il messaggio che richiede di prendere il portafoglio o di raggiungere il credito necessario per l’acquisto(10€), altrimenti restituisce “true” e acquista il biglietto , con la possibilità di riceve randomicamente un Bonus di 40 punti. Il bonus sarà gestito da una procedura (Aggiorna\_Tempo\_bonus()).

**void** **Gioco::Aggiorna\_Tempo\_bonus**() // Museo

{

**int** tempo\_b=40;

interfaccia.scrivi("---------------Complimenti hai ottenuto 40 punti bonus!----------------");

cout<<"Tempo precedente: "<<tempo<<**endl**;

tempo=tempo+tempo\_b;

cout<<"Il tempo attuale e':"<<tempo<<**endl**;

}

**bool** **Gioco::biglietto\_museo**() //funzione ausiliare per controllare la disponibilita' del portafoglio

// e aquistare biglietto museo

{ **bool** acquisto=**false**;

**char** a[1];

**int** tempo\_b=0;

**float** saldo\_p=0.0f;

**int** numero;

**bool** trovato = **false**;

trovato = portafoglio.hai\_Portafoglio(oggetti);

**if**(trovato==**true**)

{

saldo\_p=portafoglio.get\_contanti();

**if** (saldo\_p>=10)

{

saldo\_p=saldo\_p-10;

portafoglio.set\_contanti(saldo\_p);

**do**

{

**system**("cls");

interfaccia.scrivi("Biglietto acquistato!");

interfaccia.a\_capo();

//Calcolo bonus randomico

numero=**rand**()%(10-1+1)+1;

**if**(numero==3)

{

Aggiorna\_Tempo\_bonus();

}

interfaccia.scrivi("Premi un tasto per continuare...");

cin>>a;

}**while**(a==" ");

acquisto=**true**;

}

**else**{

interfaccia.scrivi("Credito insufficiente!");

}

}

**else** **if**(trovato==**false**)

{

interfaccia.scrivi("Non e' possibile acquistare se non indossi il portafoglio con almeno 10 euro al suo interno! ");

}

**return** acquisto;

}

La procedura mostra\_galleria\_a permette di esplorare la galleria a e di poter interagire con la relativa guida elettronica, ossia visualizzare i reperti e chiederne la descrizione:

**void** **Gioco::mostra\_galleria\_a**()

{

**int** risposta\_documento;

**bool** uscita\_museo;

**do**

{

**system**("cls");

uscita\_museo=**false**;

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

interfaccia.scrivi("|BENVENUTO NELLA GALLERIA A |");

interfaccia.scrivi("|In questa galleria verrai guidato nell'esplorazione del percorso Preistorico. |");

interfaccia.scrivi("|Il percorso Preistorico comprende una serie di esposizioni di notevole importanza.|");

interfaccia.scrivi("|Per maggiori descrizioni sui reperti presenti, consultare la guida elettronica : |");

interfaccia.scrivi("|1 - Neuropteris sp.-Felci fossili-Carbonifero-Francia |");

interfaccia.scrivi("|2 - Allosaurus fragilis-Scheletro di Allosauro-Giurassico-Stati Uniti d'America |");

interfaccia.scrivi("|3 - Coproliti di dinosauri - Epoche varie - Stati Uniti d'America |");

interfaccia.scrivi("|4 - Pterodactylus kochi-Rettile volante-Giurassico-Solnhofen (Germania) |");

interfaccia.scrivi("|5 - Tyrannosaurus rex-Palena (Chieti) |");

interfaccia.scrivi("|6 - PER USCIRE DALLA GUIDA ELETTRONICA |");

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

interfaccia.scrivi\_parziale("Mosse rimaste: ");

tempo--;

interfaccia.scrivi\_parziale(tempo);

interfaccia.scrivi\_parziale("\n");

interfaccia.scrivi\_parziale("Risposta: ");

cin >> risposta\_documento;

**switch** (risposta\_documento) {

**case** 1:

**system**("cls");

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

interfaccia.scrivi("|Neuropteris sp.-Felci fossili-Carbonifero-Francia |");

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

interfaccia.scrivi("|Queste piante raggiungevano un'altezza di cinque metri, con un tronco molto |");

interfaccia.scrivi("|simile a quello delle attuali palme. Il fusto era formato in parte dalle |");

interfaccia.scrivi("|basi foliari delle vecchie foglie, e possedeva radici aeree che crescevano |");

interfaccia.scrivi("|in prossimitÃ  della base. Le fronde erano di diverso tipo: alcune erano |");

interfaccia.scrivi("|dentellate, altre erano munite di foglioline arrotondate. A molte di queste |");

interfaccia.scrivi("|foglie sono stati assegnati nomi diversi, come Neuropteris o Alethopteris. |");

interfaccia.scrivi("|La prima era un tipo di foglia di grandi dimensioni, con un rachide percorso|");

interfaccia.scrivi("|da striature longitudinali. In Alethopteris, invece, le venature erano poco |");

interfaccia.scrivi("|visibili e quasi ad angolo retto, mentre il margine della foglia era |");

interfaccia.scrivi("|dentellato. |");

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

tempo--;

**system**("pause");

**break**;

**case** 2:

**system**("cls");

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

interfaccia.scrivi("|Allosaurus fragilis-Scheletro di Allosauro-Giurassico-Stati Uniti d'America |");

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

interfaccia.scrivi("|Allosaurus e' un genere estinto di grande dinosauro teropode, vissuto tra i |");

interfaccia.scrivi("|155 e i 145 milioni di anni fa, durante il periodo Giurassico. I primi |");

interfaccia.scrivi("|resti fossili furono ritrovati nel 1877, ad opera del paleontologo Othniel |");

interfaccia.scrivi("|Charles Marsh, che ribattezzÃ² i resti come Antrodemus. Essendo uno dei primi|");

interfaccia.scrivi("|dinosauri teropodi meglio conservati e piu' completi, questo animale ha piu'|");

interfaccia.scrivi("|volte attirato l'attenzione di paleontologi e amanti dei dinosauri. |");

interfaccia.scrivi("|L'Allosaurus era un predatore bipede di modeste dimensioni; il suo cranio |");

interfaccia.scrivi("|era incredibilmente robusto e compatto e armato di una moltitudine di denti.|");

interfaccia.scrivi("|La lunghezza di un esemplare adulto non doveva essere inferiore agli 8,5 |");

interfaccia.scrivi("|metri di lunghezza, anche se alcuni resti frammentari suggeriscono |");

interfaccia.scrivi("|dimensioni maggiori, con esemplari che avrebbero potuto raggiungere i |");

interfaccia.scrivi("|12 metri di lunghezza. |");

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

tempo--;

**system**("pause");

**break**;

**case** 3:

**system**("cls");

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

interfaccia.scrivi("|Coproliti di dinosauri - Epoche varie - Stati Uniti d'America |");

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

interfaccia.scrivi("|Il termine coprolite deriva dal greco kopros (sterco) e lÃ­thos (pietra) e |");

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

interfaccia.scrivi("|fossilizzato. Dallâ€™analisi di questi reperti, si possono ricavare |");

interfaccia.scrivi("|informazioni sulle abitudini alimentari e lâ€™habitat nel quale lâ€™animale |");

interfaccia.scrivi("|viveva. Per esempio, la presenza di semi, foglie, corteccia o radici indica |");

interfaccia.scrivi("|che lâ€™escremento e' stato prodotto da un erbivoro, mentre se si individuano |");

interfaccia.scrivi("|frammenti di ossa, artigli o tendini lâ€™animale era un carnivoro. |");

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

tempo--;

**system**("pause");

**break**;

**case** 4:

**system**("cls");

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

interfaccia.scrivi("|Pterodactylus kochi-Rettile volante-Giurassico-Solnhofen (Germania) |");

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

interfaccia.scrivi("|Pterodactylus e' un estinto genere di pterosauro, i cui membri sono |");

interfaccia.scrivi("|popolarmente chiamati ''pterodattili''. Attualmente, il genere contiene una |");

interfaccia.scrivi("|singola specie, Pterodactylus antiquus, che oltre ad essere la specie tipo |");

interfaccia.scrivi("|e' anche il primissimo genere di pterosauro mai rinvenuto. I principali |");

interfaccia.scrivi("|ritrovamenti di resti fossili di questo animale sono stati rinvenuti |");

interfaccia.scrivi("|principalmente, nei Calcari di Solnhofen, di Baviera, in Germania, risalenti|");

interfaccia.scrivi("|alla fine del periodo Giurassico, circa 150,8-148-500 milioni di anni fa, |");

interfaccia.scrivi("|anche se alcuni resti frammentari sono stati rinvenuti anche in altre aree |");

interfaccia.scrivi("|in Europa e in Africa. Questo animale era un predatore che probabilmente si |");

interfaccia.scrivi("|cibava soprattutto di pesci e piccoli invertebrati marini. Come tutti gli |");

interfaccia.scrivi("|pterosauri, anche le ali dello Pterodactyluserano formate da una membrana |");

interfaccia.scrivi("|di pelle che si estendeva dalla fine del quarto dito della ''mano'' fino |");

interfaccia.scrivi("|agli arti posteriori. L'ala era supportata, ulteriormente, internamente da |");

interfaccia.scrivi("|fibre di collagene ed esternamente da strutture cheratinose. |");

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

tempo--;

**system**("pause");

**break**;

**case** 5:

**system**("cls");

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

interfaccia.scrivi("|Pterodactylus kochi-Rettile volante-Giurassico-Solnhofen (Germania) |");

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

interfaccia.scrivi("|Il tirannosauro era un dinosauro vissuto nel Cretaceo superiore |");

interfaccia.scrivi("|appartenente alla famiglia dei tirannosauridi. Visse in nordamerica, che |");

interfaccia.scrivi("|anticamente era un continente isolato nominato Laramidia. Il Tyrannosaurus |");

interfaccia.scrivi("|era molto piu' diffuso geograficamente degli altri tirannosauridi. I suoi |");

interfaccia.scrivi("|fossili si trovano in una varietÃ  di formazioni risalenti all' epoca |");

interfaccia.scrivi("|Maastrichtiana del Cretaceo superiore, circa 68-66 milioni di anni fa. |");

interfaccia.scrivi("|Fu una delle specie degli ultimi dinosauri non-aviani viventi quando si ebbe|");

interfaccia.scrivi("|l' estinzione di massa del Cretaceo-Paleocene, che determino' la scomparsa |");

interfaccia.scrivi("|dei dinosauri propriamente detti. |");

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

tempo--;

**system**("pause");

**break**;

**case** 6:

uscita\_museo=**true**;

**system**("cls");

**break**;

**default**:

interfaccia.scrivi("Comando non riconosciuto");

**system**("pause");

**break**;

}

}**while**(uscita\_museo!=**true**);

}

La stessa procedura vale per la mostra delle gallerie b,c,d richiamate all’interno dello switch dell’azione 96:

**void** **Gioco::mostra\_galleria\_b**()

{**bool** uscita\_museo;

**do**

{

uscita\_museo=**false**;

**system**("cls");

**int** risposta\_documento;

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

interfaccia.scrivi("|BENVENUTO NELLA GALLERIA B |");

interfaccia.scrivi("|In questa galleria verrai guidato nell'esplorazione del percorso Romano. |");

interfaccia.scrivi("|Come testimonianza di questo periodo storico potrete ammirare antichi affreschi |");

interfaccia.scrivi("|Per ulteriri 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 cittÃ . |");

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

tempo--;

**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 citta' 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 â€œa falsa portaâ€� 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 dellâ€™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 Urkaptah, 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 â€œfalsa portaâ€�, |");

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("|lâ€™iscrizione in caratteri geroglifici sottostante completa per forza magica |");

interfaccia.scrivi("|Lâ€™importanza del pane, quale alimento base della dieta egiziana, appare |");

interfaccia.scrivi("|evidente, anche se non e' possibile differenziare le varie tipologie di |");

interfaccia.scrivi("|pane elencate per qualitÃ  di ingredienti, modalitÃ  di impasto e di cottura |");

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

tempo--;

**system**("pause");

**break**;

**case** 2:

**system**("cls");

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

interfaccia.scrivi("|Rilievo con scena di libagione |");

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

interfaccia.scrivi("|Il rilievo, collocato in origine alle pareti di una tomba, mostra una scena |");

interfaccia.scrivi("|di libagione. La â€œsignora della casaâ€� 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 allâ€™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 â€œanimaâ€� 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 dellâ€™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("|dâ€™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 funararia |");

interfaccia.scrivi("|dell'Antico Regno, che rende il sarcofago simile a una dimora eterna. Su uno |");

interfaccia.scrivi("|dei lati lunghi della cassa, in posizione centrale, sono raffigurate numerose |");

interfaccia.scrivi("|offerte alimentari (pani, ortaggi, frutti, tranci di carni bovine, volatili |");

interfaccia.scrivi("|giÃ  spennati, contenitori per liquidi, etc.), perche' il defunto possa |");

interfaccia.scrivi("|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 civilta' 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 nuditÃ  eroica. |");

interfaccia.scrivi("|3 - Statua marmorea del II secolo d.C. di genio carpoforo (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 nellâ€™atto di offrire una melagrana ad un serpente, simbolo |");

interfaccia.scrivi("|funerario delle divinitÃ  infernali. |");

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

tempo--;

**system**("pause");

**break**;

**case** 3:

**system**("cls");

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

interfaccia.scrivi("|Statua marmorea del II secolo d.C. di genio carpoforo (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");

}

### 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.

### 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*

## Aula singola

Sono state effettuate modifiche nei seguenti file:

* Astro.cpp
* Astro.h
* Mappa.nav

### 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

### 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

### 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

#### Implementazione azione\_85

**void** **Astro::azione\_85**(){

interfaccia.scrivi("Ti sei seduto");

storia\_gioco.insStoria(stringa\_comando , "Hai deciso di sederti"); //Aggiorna storia gioco

}

#### 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 randomizzera un numero da 1 a 3 che servirà per decidere quale libro leggere

**if** (lib==1 && controllo1==**true**){// se il libro e' stato gia' letto

**while**(l.finelista(p)!=**true**){//fino a quando non raggiungiamo finelista

**if**(l.leggilista(p)==1){

interfaccia.scrivi("È UN LIBRO DI MANZONI");

interfaccia.scrivi("L'HAI GIA' LETTO!!");

}

p=l.succlista(p);

}

}

**else** **if** (lib==1 && controllo1==**false**) {//se il libro non è stato ancora letto

interfaccia.scrivi("È UN LIBRO DI ALLESSANDRO MANZONI");

interfaccia.scrivi("");

interfaccia.scrivi("Ei fu. Sicome immobile,");

interfaccia.scrivi("dato il mortal sospiro,");

interfaccia.scrivi("stette la spoglia immemore ");

interfaccia.scrivi("orba di tanto spiro,...");

l.inslista(1,p);// inserisce nella lista il numero 1

p=l.succlista(p);

controllo1=**true**;//metto a true il controllo della lettura del libro

**int** salute=Salute.GetStatoSalute();// creo una variabile salute in cui metterò il valore della salute del giocatore

**if**(salute>=95 ){// se la salute è maggiore o uguale di 95 e non mai stato utilizzato il kit medico

interfaccia.scrivi("");

interfaccia.scrivi("HAI TROVATO UN KIT MEDICO MA LA TUA SALUTE È ANCORA OTTIMA ");

interfaccia.scrivi("LO USI MA NON HA NESSUN EFFETTO");

}

**else** {// se la salute non è piu maggiore di 95 e non è mai stato utilizzato il kit medico

interfaccia.scrivi("");

Salute.SetStatoSalute(salute+5);// agiorno la salute aggiungendo 5 unità

interfaccia.scrivi("INCREDIBILE!!");

interfaccia.scrivi("HAI RECUPERATO CINQUE UNITA 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 e' stato gia' letto

**while**(l.finelista(p)!=**true**){//fino a quando non raggiungiamo finelista

**if**(l.leggilista(p)==2){

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

interfaccia.scrivi("L'HAI GIA' LETTO!!");

}

p=l.succlista(p);

}

}

**else** **if** (lib==2 && controllo2==**false**) {//se il libro non è stato ancora letto

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

interfaccia.scrivi("");

interfaccia.scrivi("M'illumino d'immenso");

l.inslista(2,p);// inserisce nella lista il numero 2

p=l.succlista(p);

controllo2=**true**;//metto a true il controllo della lettura del libro

}

**else** **if**(lib==3 && controllo3==**true**){// se il libro e' stato gia' letto

**while**(l.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 GIA' LETTO!!");

}

p=l.succlista(p);

}

}

**else** **if** (lib==3 && controllo3==**false**) {//se il libro non è stato ancora letto

interfaccia.scrivi("È UN LIBRO DI GIACOMO LEOPARDI");

interfaccia.scrivi("");

interfaccia.scrivi("Silvia, rimembri ancora");

interfaccia.scrivi("quel tempo della tua vita mortale,");

interfaccia.scrivi("quando belta' splendea...");

l.inslista(3,p);// inserisce nella lista il numero 3

p=l.succlista(p);

controllo3=**true**;//metto a true il controllo della lettura del libro

}

storia\_gioco.insStoria(stringa\_comando , "hai deciso di leggere dei libri"); //Aggiorna storia gioco

}

#### Implementazione azione\_87

**void** **Astro::azione\_87**(){

**int** num=(**rand**() % 4) + 1;// creo una variabile che mi randomizzera 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 semplicemnte, numeri relativi) ");

interfaccia.scrivi("sono formati dall'unione dei numeri naturali (0,1,2,..) e dei numeri interi negativi (-1,-2,-3,..) ");

interfaccia.scrivi("costruiti ponendo un segno - davanti ai numeri naturali");

}

**else** **if**(num==2){

interfaccia.scrivi("Ricerca di geografia:");

interfaccia.scrivi("Taranto e' un comune italiano di 202.063 abitanti capoluogo dell'omonima provincia, in Puglia");

interfaccia.scrivi("Il clima e' dolce e mite.");

interfaccia.scrivi("Taranto inoltre e' la casa della marina militare italiana.");

}

**else** **if**(num==3){

interfaccia.scrivi("Ricerca di storia:");

interfaccia.scrivi("La prima guerra mondiale fu un conflitto armato");

interfaccia.scrivi("che coinvolse le principali potenze mondiali e molte di quelle minori. ");

interfaccia.scrivi("Il conflitto duro' dal 28 luglio 1914 e l'11 novembre 1918.");

}

**else** **if**(num==4){

interfaccia.scrivi("Ricerca di informatica:");

interfaccia.scrivi("Un bit e' l'unita di misura dell'informazione(binary digit),");

interfaccia.scrivi("definita come la quantita' minima di informazione");

interfaccia.scrivi("che serve a discernere tra due possibili ");

interfaccia.scrivi("eventi equiprobabili.");

}

storia\_gioco.insStoria(stringa\_comando , "hai deciso di usare il computer"); //Aggiorna storia gioco

}

#### 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'obbiettivo di una attivita' lavorativa");

interfaccia.scrivi("senza l'esposizione per i lavoratori al rischio di infortuni/incidenti");

interfaccia.scrivi("e senza il richio 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 posseso 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 sopratutto 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

}

#### 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

}

#### 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

}

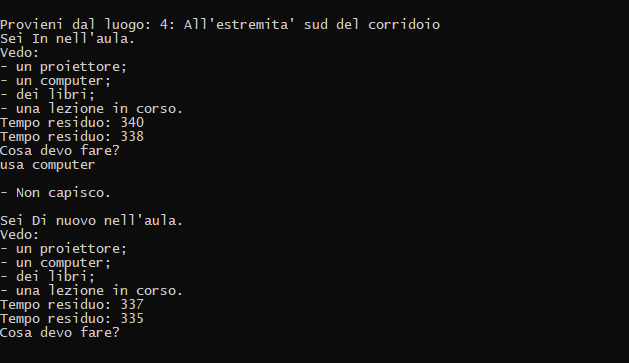
### Implementazione dell’Aula

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*

### Correzione Funzionalità usa computer (Andrea Tursi)

Nel luogo aula la funzionalità “usa computer” non veniva riconosciuta durante le fasi di gioco.



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.

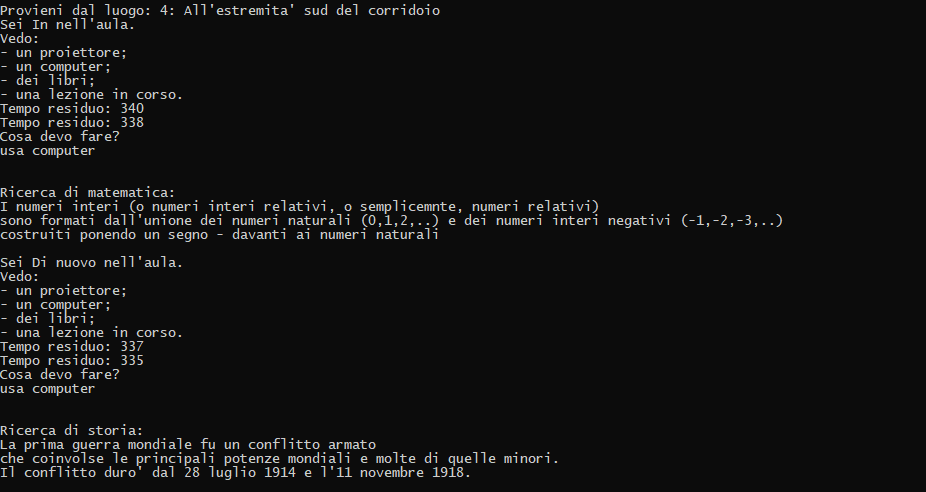
Allora si è pensato di evitare l’inserimento di un secondo vocabolo “computer”, eliminando la riga di codice apposita, e di modificare il file Astro.cpp in modo tale far riferimento al vocabolo già presente.

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"



## Ufficio Postale

### Inserimento del luogo "Ufficio Postale"

#### 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**

#### 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"

### Gioco.h

Riga 44 :

**#include** "UfficioPostale.h"

Riga 161 :

UfficioPostale ufficiopostale;

### Codice per inserimento oggetti

Inserimento vocaboli :

#### 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 :

#### 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.

### 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.

#### Astro.h

**void** **azione\_82**();//Guarda Ufficio Postale

**void** **azione\_83**();//Apri Pacco

**void** **azione\_84**();//Leggi Lettera

#### 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 ogetto 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.");

}

}

### Modifiche al progetto di Bellanova

Oltre alle aggiunte fatte al progetto di D'Andria\Dresda sono state necessare 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

## Dialoghi

### Implementazione delle variabili

#### Personaggi.h

Includiamo la classe Dialogo

|  |
| --- |
| 1. #include "Dialogo.h" //Aggiunta D'Andria Dresda al progetto di Federica Forte |

Aggiungiamo l'attributo dialogo ai personaggi

|  |
| --- |
| 1. Dialogo dialogo; //Aggiunta D'Andria Dresda al progetto di Federica Forte |

### Implementazione dei metodi

#### Personaggi.h

Aggiungiamo il metodo per settare un dialogo ad un personaggio

|  |
| --- |
| 1. **void** setDialogo(Dialogo\*); //aggiunta D'Andria Dresda al progetto di Federica Forte |

Aggiungiamo il metodo per prendere il dialogo di un personaggio

|  |
| --- |
| 1. Dialogo\* getDialogo(); //aggiunta D'Andria Dresda al progetto di Federica Forte |

#### Personaggi.cpp

##### Metodo setDialogo:

|  |
| --- |
| 1. **void** Personaggi::setDialogo(Dialogo \*dialogodainserire){ 2. **this**->dialogo=\*dialogodainserire; 3. } |

##### Metodo getDialogo:

|  |
| --- |
| 1. Dialogo\* Personaggi::getDialogo(){ 2. **return** (&**this**->dialogo); 3. } |

#### Gioco.h

|  |
| --- |
| 1. Dialogo caricaDialoghiDaFile(string); 2. **void** convertiRiga(Dialogo\*,string,string); 3. **int** caricastatodialogo(string); 4. **int** leggistato(**int**\*,string, string); 5. **void** scrivistatosufile(string, **int**); 6. **bool** verificanome(string,string); 7. **void** salvaStatiDialoghi(); 8. **void** caricaStatiDialoghi(); |

#### Gioco.cpp

##### 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.

|  |
| --- |
| 1. Dialogo Gioco::caricaDialoghiDaFile(string nome){ 2. Dialogo dialogo= Dialogo(nome); 3. string rigafile; 4. ifstream filedialoghi("Dialoghi.txt", ios::in); 5. **while** (filedialoghi.get()!='?'){ 6. getline(filedialoghi,rigafile); 7. convertiRiga(&dialogo,nome,rigafile); 8. } 9. filedialoghi.close(); 10. **return** dialogo; 11. } |

##### 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$0:interrompi dialogo$#*

###### 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.

|  |
| --- |
| 1. **void** Gioco::convertiRiga(Dialogo \*dialogo,string nome,string rigafile){ 2. string nomepersonaggio=""; 3. **char** statodialogostr[20]; 4. **for**(**int** j=0; j<20; j++) 5. statodialogostr[j]='\0'; 6. **int** statodialogo; 7. string frasedialogo=""; 8. **int** indice=0; 9. **while**(rigafile[indice]!='|'){ 10. nomepersonaggio=nomepersonaggio+rigafile[indice]; //leggo il nome del personaggio 11. indice++; 12. } 13. indice++; 14. **int** i=0; 15. **while**(rigafile[indice]!='\*'){ 16. statodialogostr[i]=rigafile[indice]; // leggo lo stato del dialogo 17. indice++; 18. i++; 19. } 20. statodialogo=atoi(statodialogostr); 21. indice++; 22. **while**(rigafile[indice]!='#'){ 23. **if**(rigafile[indice]=='$') 24. frasedialogo=frasedialogo+'\n'; 25. **else** 26. frasedialogo=frasedialogo+rigafile[indice]; // leggo la frase da inserire nel nodo Domanda 27. indice++; 28. } 29. **if** (nome==nomepersonaggio) 30. dialogo->inseriscidomanda(statodialogo,frasedialogo); //inserisce la Domanda nel Dialogo 31. } |

#### 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 '.

|  |
| --- |
| 1. **int** Gioco::caricastatodialogo(string nome){ 2. **int** stato=1; 3. string rigafile; 4. ifstream filestati("Statidialoghi.txt", ios::in); 5. **if**(!filestati.is\_open()) 6. cout << "File Statidialoghi.txt non trovato"; 7. **else**{ 8. **while** (filestati.get()!='?'){ 9. getline(filestati,rigafile); 10. leggistato(&stato,nome,rigafile); //legge lo stato all'interno della riga del file 11. } 12. } 13. filestati.close(); 14. **return** stato; 15. } |

#### 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 '.

|  |
| --- |
| 1. **int** Gioco::leggistato(**int** \*stato,string nome, string rigafile){ 2. string nomepersonaggio; 3. **char** statodialogostr[20]; 4. **for**(**int** j=0; j<20; j++) 5. statodialogostr[j]=NULL; 6. **int** statodialogo; 7. **int** indice=0; 8. **while**(rigafile[indice]!='|'){ 9. nomepersonaggio=nomepersonaggio+rigafile[indice]; 10. indice++; 11. } 12. indice++; 13. **int** i=0; 14. **while**(rigafile[indice]!='#'){ 15. statodialogostr[i]=rigafile[indice]; 16. indice++; 17. i++; 18. } 19. statodialogo=atoi(statodialogostr); 20. **if** (nome==nomepersonaggio) 21. \*stato=statodialogo; 22. } |

#### 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 '

|  |
| --- |
| 1. **void** Gioco::scrivistatosufile(string nome, **int** stato){ 2. ifstream file("Statidialoghi.txt"); 3. ofstream nuovofile("Nuovofile.txt"); 4. string rigafile; 5. string nuovariga; 6. **bool** riganome=**false**; 7. **if**(!file) 8. cout << "File Statidialoghi.txt non trovato"; 9. **else**{ 10. **while**(file.get()!='?'){ 11. getline(file,rigafile); 12. riganome=verificanome(nome,rigafile); 13. **if**(riganome){ 14. stringstream statostream; 15. statostream<<stato; 16. string statostring=statostream.str(); 17. nuovariga='-'+nome+'|'+statostring+'#'; 18. nuovofile << nuovariga << '\n'; 19. } 20. **else** 21. nuovofile << '-' << rigafile << '\n'; 22. } 23. } 24. nuovofile << '?'; 25. file.close(); 26. nuovofile.close(); 27. std::remove("Statidialoghi.txt"); 28. **int** result=rename("Nuovofile.txt","Statidialoghi.txt"); 29. } |

#### 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.

|  |
| --- |
| 1. **bool** Gioco::verificanome(string nome, string rigafile){ 2. string nomepersonaggio; 3. **int** indice=0; 4. **while**(rigafile[indice]!='|'){ 5. nomepersonaggio=nomepersonaggio+rigafile[indice]; 6. indice++; 7. } 8. **return**(nome==nomepersonaggio); 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.

|  |
| --- |
| 1. **void** Gioco::salvaStatiDialoghi(){ 2. ofstream nuovofile("Astrostatidialoghi.txt"); 3. ifstream file("Statidialoghi.txt"); 4. string rigafile; 5. **while**(file.get()!='?'){ 6. getline(file,rigafile); 7. nuovofile << '-' << rigafile << '\n'; 8. } 9. nuovofile << '?'; 10. file.close(); 11. nuovofile.close(); 12. } |

#### 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.

|  |
| --- |
| 1. **void** Gioco::caricaStatiDialoghi(){ 2. ifstream file("Astrostatidialoghi.txt"); 3. ofstream nuovofile("Statidialoghi.txt"); 4. string rigafile; 5. **while**(file.get()!='?'){ 6. getline(file,rigafile); 7. nuovofile << '-' << rigafile << '\n'; 8. } 9. nuovofile << '?'; 10. file.close(); 11. nuovofile.close(); 12. } |

### 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.

#### Save

|  |
| --- |
| 1. **void** Gioco::load(){ 2. **int** i; 3. **int** valore; 4. **int** slot; //CICALA GIACOMO 5. svuotaInsieme(slotmachine); //CICALA GIACOMO 7. ifstream file(fStringa.c\_str(), ios::in); 8. //modifiche effettuate da D'Andria Dresda sul controllo del caricamento da file 9. ifstream astrostatidialoghi("Astrostatidialoghi.txt"); 10. **bool** astrostatidialoghiaperto=astrostatidialoghi.good(); 11. **bool** fileaperto=file.good(); 12. **if**(fileaperto || astrostatidialoghiaperto){ 13. interfaccia.scrivi("Ripristino partita..."); 14. **if**(fileaperto){ 15. **for** (i = 1; i <= oggetti.get\_n\_oggetti(); i++){ 16. file >> valore; 17. oggetti.set\_luogo(i,valore); 18. } 19. file >> luogo\_attuale; 20. file >> tempo; 21. file >> passo\_soluzione; 22. load\_specifiche(file); 23. bacheca.CaricaBacheca(file); 24. file.close(); 25. } 26. **if**(astrostatidialoghiaperto) 27. caricaStatiDialoghi(); //modifiche D'Andria Dresda 28. ifstream jukeLuci("JukeLuci.txt"); 29. **if** (jukeLuci.good() ) 30. caricaJukeBoxLuci(); 31. } 32. **else** 33. interfaccia.scrivi("Non ci sono partite salvate!"); 34. //INIZIO MODIFICHE CICALA GIACOMO 35. file >> numEuro; 36. **for**(**int** j=1; j<=numEuro; j++){ 37. file >> slot; 38. slotmachine.inserisci(slot); 39. } 40. //fine modifiche CICALA GIACOMO 41. } |

#### switch\_enigmi

|  |
| --- |
| 1. // modifica zagaria -- aggiunto il paramentro per la gestione dei personaggi 2. **void** Gioco::switch\_enigmi(**int** a, Mappa &M, Pila<stato\_comando>\* p, Personaggi\* pg){ 3. **bool** parlato; //modifica D'Andria Dresda al progetto di Federica Forte 4. **if**(leggienigma()){ 5. cout<<"\nComplimenti hai indovinato l'enigma puoi proseguire" << endl; 6. cout<<"\n"; 7. //Modifica Pmf: fin qui. 8. //INIZIO MODIFICA MICHELE ALBANO 9. **if**(!sFisico.feritaEvitata(rand()))//Modifica DAVIDE MANTELLINI 10. Aggiorna\_Ferite(); //Generatore Ferite 11. **if**(Salute.GetStatoSalute()==0) //Se lo Stato della Salute è a 0, fine dell'avventura 12. morto(); 13. interfaccia.a\_capo(); 14. //FINE MODIFICA MICHELE ALBANO 15. **switch** (a){ 16. **case** 1: 17. parlato=**false**; //aggiunta D'Andria Dresda dal progetto di Gallo in quello di Federica Forte 18. direzioni(M); 19. **break**; 20. **case** 2: 21. prendi(); 22. **break**; 23. **case** 3: 24. lascia(); 25. **break**; 26. **case** 4: 27. guarda(); 28. **break**; 29. **case** 5: 30. save(); 31. **break**; 32. **case** 6: 33. load(); 34. **break**; 35. **case** 7: 36. cosa(); 37. **break**; 38. **case** 8: 39. navigatore(M); 40. system("cls"); 41. //system("clear"); //linux os  //Modificato da ANTONIO PASTORELLI 42. **break**; 43. **case** 9: 44. indietro(p); 45. **break**; 46. **case** 39: 47. macchinadeltempo(); 48. **break**; 49. //aggiunta D'Andria Dresda dal progetto di Federica Forte 50. **case** 99: 51. **if**(parlato==**true**) 52. cout<<"Ci hai gia' parlato"<<endl; 53. **else** 54. { 55. //modifica D'Andria Dresda 56. az\_parla(\*pg); 57. **if**(pg->getNome()!="Cid" && pg->getNome()!="Vincent" && pg->getNome()!="Kail") 58. parlato=**true**; 59. } 60. **break**; 61. //fine aggiunta D'Andria Dresda 62. **case** 40: 63. visualizza\_bacheca(); 64. system("cls"); 65. //system("clear"); //linux os  //Modificato da ANTONIO PASTORELLI 66. **break**; 67. **case** 41: 68. benzina(); 69. **break**; 70. **case** 42: 71. usa\_motorino(); 72. **break**; 73. // inizio modifiche zagaria 74. **case** 43: 75. stringa\_risposta = "Hai parlato con il personaggio presente nella stanza."; //aggiungo alla storia 76. storia\_gioco.insStoria(stringa\_comando, stringa\_risposta); 77. az\_parla(\*pg); 78. **break**;                          //fine modifiche 79. //Modifica PMF(agenda) 80. **case** 46: 81. azione\_46(); 82. **break**; 83. **case** 47: 84. azione\_47(); 85. **break**; 86. **case** 48: 87. azione\_48(); 88. **break**; 89. //Modifica PMF: fin qui. 90. //START DAMONE 91. //Modifica Francesco De Giorgio 92. **case** 49: 93. consulta();//inserita la consultazione della guida 94. //Modifica De Giorgio : fin qui. 95. **break**; 96. //END DAMONE 97. **default**: // AZIONI SPECIFICHE 98. **if** (!esegui\_specifiche(a,M)) 99. interfaccia.scrivi("AZIONE " + a); // test 100. } 101. } 102. //Modifica PMF(enigmi) 103. **else**{ 104. cout << "\nHai sbagliato la risposta stai perdendo tempo," << endl; 105. cout <<"ridigita il comando per avere un altro enigma" << endl; 106. tempo--; 107. } 108. } |

#### esegui

|  |
| --- |
| 1. // MODIFICA zagaria -- modifica della funzione esegui con l' aggiunta del parametro personaggi 2. **void** Gioco::esegui(**int** a, Mappa &M, Pila<stato\_comando>\* p, **bool** si, Personaggi\* pg){ 4. **if**(si) 5. switch\_enigmi(a,M,p,pg); // modifica zagaria -- aggiunto il parametro pg 6. **else**{ 7. **bool** parlato; //Aggiunta D'Andria Dresda 8. //INIZIO MODIFICA MICHELE ALBANO 9. Aggiorna\_Ferite(); //Generatore Ferite 10. **if**(Salute.GetStatoSalute()==0) //Se lo Stato della Salute è a 0, fine dell'avventura 11. morto(); 12. interfaccia.a\_capo(); 13. //FINE MODIFICA MICHELE ALBANO 14. **switch** (a){ 15. **case** 1: 16. parlato = **false**; //Aggiunta D'Andria Dresda 17. direzioni(M); 18. **break**; 19. **case** 2: 20. prendi(); 21. **break**; 22. **case** 3: 23. lascia(); 24. **break**; 25. **case** 4: 26. guarda(); 27. **break**; 28. **case** 5: 29. save(); 30. **break**; 31. **case** 6: 32. load(); 33. **break**; 34. **case** 7: 35. cosa(); 36. **break**; 37. **case** 8: 38. navigatore(M); 39. system("cls"); 40. //system("clear"); //linux os  //Modificato da ANTONIO PASTORELLI 41. **break**; 42. **case** 9: 43. indietro(p); 44. **break**; 45. **case** 39: 46. macchinadeltempo(); 47. **break**; 48. //aggiunta D'Andria Dresda 49. **case** 99: 50. **if**(parlato==**true**) 51. cout<<"Ci hai gia' parlato"<<endl; 52. **else**{ 53. az\_parla(\*pg); 54. **if**(pg->getNome()!="Cid" && pg->getNome()!="Vincent" && pg->getNome()!="Kail") //modifica D'Andria Dresda 55. parlato=**true**; 56. } 57. **break**; 58. //fine aggiunta D'Andria Dresda 59. **case** 40: 60. visualizza\_bacheca(); 61. system("cls"); 62. //system("clear"); //linux os  //Modificato da ANTONIO PASTORELLI 63. **break**; 64. **case** 41: 65. benzina(); 66. **break**; 67. **case** 42: 68. usa\_motorino(); 69. **break**; 70. // inizio modifiche zagaria -- definisce la comunicazione con il personaggio nella storia 71. **case** 43: 72. stringa\_risposta = "Hai parlato con il personaggio presente nella stanza."; //aggiungo alla storia 73. storia\_gioco.insStoria(stringa\_comando, stringa\_risposta); 74. az\_parla(\*pg); 75. **break**; 76. //fine modifiche 77. //Modifica PMF(agenda) 78. **case** 46: 79. azione\_46(); 80. **break**; 81. **case** 47: 82. azione\_47(); 83. **break**; 84. **case** 48: 85. azione\_48(); 86. **break**; 87. //Modifica PMF: fin qui. 88. //START DAMONE 89. //Modifica Francesco De Giorgio 90. **case** 49: 91. consulta();//inserita la consultazione della guida 92. //Modifica De Giorgio : fin qui. 93. **break**; 94. //END DAMONE 95. **default**: // AZIONI SPECIFICHE 96. **if** (!esegui\_specifiche(a,M)) 97. interfaccia.scrivi("AZIONE " + a); // test 98. } 99. } 100. } |

#### initPers

|  |
| --- |
| 1. **void** Gioco::initPers(Dizionario<**int**, Personaggi> \*insPersonaggi) 2. { 4. 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\*/, 5. 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,\*/ 6. 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,\*/ 7. p91,p92,p93/\*,p94,p95,p96,p97,p98,p99,p100\*/; 9. //AGGIUNTA DEI PERSONAGGI PER I DIALOGHI BASILE ANTONIO 10. Personaggi prof\_Mara, prof\_Clara, alunno\_Davide; 12. //Luogo 1 1-10 14. p1.setNome("Kain"); 15. p1.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 16. p1.setLuogo(1); 17. insPersonaggi->inserisci(1, p1); 18. p2.setNome("Tom"); 19. p2.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 20. p2.setOgg(**true**,51); 21. p2.setFrasiconOgg1("Visto che hai la tuta puoi andare, qui non c'e' altro di importante","Controla tutti gli indicatori e i pulsanti, potrebbero mostrarti qualcosa di utile"); 22. p2.setLuogo(1); 23. insPersonaggi->inserisci(2, p2); 24. p3.setNome("Jim"); 25. p3.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 26. p3.setLuogo(1); 27. insPersonaggi->inserisci(3, p3); 28. p4.setNome("Adam"); 29. p4.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 30. p4.setOgg(**true**,51); 31. p4.setFrasiconOgg1("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"); 32. p4.setLuogo(1); 33. insPersonaggi->inserisci(4, p4); 34. p5.setNome("Paul"); 35. p5.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 36. p5.setLuogo(1); 37. insPersonaggi->inserisci(5, p5); 38. p6.setNome("Raoul"); 39. p6.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 40. p6.setOgg(**true**,51); 41. p6.setFrasiconOgg1("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"); 42. p6.setLuogo(1); 43. insPersonaggi->inserisci(6, p6); 44. p7.setNome("Max"); 45. p7.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 46. p7.setLuogo(1); 47. insPersonaggi->inserisci(7, p7); 48. p8.setNome("Sam"); 49. p8.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 50. p8.setOgg(**true**,51); 51. p8.setFrasiconOgg1("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"); 52. p8.setLuogo(1); 53. insPersonaggi->inserisci(8, p8); 54. p9.setNome("Yan"); 55. p9.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 56. p9.setLuogo(1); 57. insPersonaggi->inserisci(9, p9); 58. p10.setNome("Arthur"); 59. p10.setFrasi("Prendi la 'Tuta' potrebbe servirti","La 'Tuta' e' inutile, lasciala li"); 60. p10.setOgg(**true**,51); 61. p10.setFrasiconOgg1("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"); 62. p10.setLuogo(1); 63. insPersonaggi->inserisci(10, p10); 65. //Luogo 2 11-20 66. p11.setNome("Cloud"); 67. p11.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'"); 68. p11.setLuogo(2); 69. insPersonaggi->inserisci(11, p11); 70. p12.setNome("Cole"); 71. p12.setFrasi("Controlla la 'Cabina del secondo pilota', potresti trovare qualcosa di utile","La 'Cabina del secondo pilota' e vuota, e inutile entrarci"); 72. p12.setLuogo(2); 73. insPersonaggi->inserisci(12, p12); 74. p13.setNome("Zack"); 75. p13.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'"); 76. p13.setOgg(**true**,55); 77. p13.setFrasiconOgg1("Hai il 'manuale',il reattore deve essere spento, svelto","Il 'manuale' non ti servira' a nulla, lascialo e cerca una soluzione al problema"); 78. p13.setLuogo(2); 79. insPersonaggi->inserisci(13, p13); 80. p14.setNome("Getro"); 81. p14.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'"); 82. p14.setLuogo(2); 83. insPersonaggi->inserisci(14, p14); 84. p15.setNome("Leroy"); 85. p15.setFrasi("Controlla la 'Cabina del secondo pilota', potresti trovare qualcosa di utile","La 'Cabina del secondo pilota' e' vuota, e inutile entrarci"); 86. p15.setLuogo(2); 87. insPersonaggi->inserisci(15, p15); 88. p16.setNome("Tony"); 89. p16.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'"); 90. p16.setOgg(**true**,55); 91. p16.setFrasiconOgg1("Hai il 'manuale',il reattore deve essere spento, svelto","Il 'manuale' non ti servira' a nulla, lascialo e cerca una soluzione al problema"); 92. p16.setLuogo(2); 93. insPersonaggi->inserisci(16, p16); 94. p17.setNome("Timmy"); 95. p17.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'"); 96. p17.setLuogo(2); 97. insPersonaggi->inserisci(17, p17); 98. p18.setNome("Eric"); 99. p18.setFrasi("Controlla la 'Cabina del secondo pilota', potresti trovare qualcosa di utile","La 'Cabina del secondo pilota' e vuota, e inutile entrarci"); 100. p18.setLuogo(2); 101. insPersonaggi->inserisci(18, p18); 102. p19.setNome("Dan"); 103. p19.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'"); 104. p19.setOgg(**true**,55); 105. p19.setFrasiconOgg1("Hai il 'manuale',il reattore deve essere spento, svelto","Il 'manuale' non ti servira' a nulla, lascialo e cerca una soluzione al problema"); 106. p19.setLuogo(2); 107. insPersonaggi->inserisci(19, p19); 108. p20.setNome("Chuck"); 109. p20.setFrasi("La 'tuta' dovrebbe trovarsi nella 'Cabina di pilotaggio'","La 'tuta' si trova nella 'sala di controllo del reattore'"); 110. p20.setLuogo(2); 111. insPersonaggi->inserisci(20, p20); 113. //luogo 3 21-30 114. p21.setNome("Vincent"); 115. p21.setFrasi("A nord trovi la 'Cabina di pilotaggio'","A sud trovi la 'Cabina di pilotaggio'"); 116. p21.setLuogo(3); 117. //modifiche D'Andria Dresda aggiunta dialogo a Vincent 118. Dialogo dialogovincent=Dialogo("Vincent"); 119. dialogovincent=caricaDialoghiDaFile("Vincent"); 120. p21.setDialogo(&dialogovincent); 122. //fine modifiche 123. insPersonaggi->inserisci(21, p21); 124. p22.setNome("Kail"); 125. p22.setFrasi("A sud trovi 'il compartimento stagno' e la 'sala del reattore'","A sud trovi la 'Cabina di pilotaggio'"); 126. p22.setLuogo(3); 127. //modifiche D'Andria Dresda aggiunta dialogo a Vincent 128. Dialogo dialogokail=Dialogo("Kail"); 129. dialogokail=caricaDialoghiDaFile("Kail"); 130. p22.setDialogo(&dialogokail); 131. //fine modifiche 133. insPersonaggi->inserisci(22, p22); 135. p23.setNome("Joe"); 136. p23.setFrasi("Cerca il 'casco'","Il 'casco' e inutile"); 137. p23.setOgg(**true**,50,51); 138. p23.setFrasiconOgg1("Cerca la 'tuta'","la 'tuta' non ti serve"); 139. p23.setFrasiconOgg2("Cerca il 'secondo pilota'","Il 'secondo pilota' e nella sua cabina"); 140. p23.setLuogo(3); 141. insPersonaggi->inserisci(23, p23); 143. //luogo 4 31-40 144. p31.setNome("Garnet"); 145. p31.setFrasi("Ho visto il 'secondo pilota' andare verso il 'compartimento stagno' a est","Il secondo pilota non e' passato di qui"); 146. p31.setLuogo(4); 147. insPersonaggi->inserisci(31, p31); 148. p32.setNome("Shaun"); 149. p32.setFrasi("Ho visto il 'secondo pilota' andare verso il 'compartimento stagno' a est","Il secondo pilota non e' passato di qui"); 150. p32.setOgg(**true**,50,51); 151. p32.setFrasiconOgg1("Cerca la 'tuta' prima di uscire dall'astronave","Corri nel 'compartimento stagno' e cerca il 'secondo pilota'"); 152. p32.setFrasiconOgg2("Corri nel 'compartimento stagno' e cerca il 'secondo pilota'","Lascia la 'tuta' prima di andare"); 153. p32.setLuogo(4); 154. insPersonaggi->inserisci(32, p32); 155. p33.setNome("Frank"); 156. p33.setFrasi("Ho visto il 'secondo pilota' andare verso il 'compartmento stagno' a est","Il secondo pilota non e' passato di qui"); 157. p33.setOgg(**true**,54); 158. p33.setFrasiconOgg1("Vai nella 'cabina del secondo pilota'","La 'Chiave' che hai preso non va bene, cercane un altra"); 159. p33.setLuogo(5); 160. insPersonaggi->inserisci(33, p33); 162. //luogo 5 41-50 163. p41.setNome("Cecil"); 164. p41.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo","Nell''Armadietto' non c'e nulla"); 165. p41.setLuogo(5); 166. insPersonaggi->inserisci(41, p41); 167. p42.setNome("Vivi"); 168. p42.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo","Nell''Armadietto' non c'e nulla"); 169. p42.setOgg(**true**,54); 170. p42.setFrasiconOgg1("Apri l''Armadietto', potresti trovare qualcosa di utile","La 'Chiave' che hai preso non va bene per questo armadietto, cercane un altra"); 171. p42.setLuogo(5); 172. insPersonaggi->inserisci(42, p42); 173. p43.setNome("Jin"); 174. p43.setFrasi("La 'Chiave' la possiede il 'secondo pilota', cercalo","Nell''Armadietto' non c'e nulla"); 175. p43.setOgg(**true**,54); 176. p43.setFrasiconOgg1("Togli la 'tata' prima di prendere il camice","Il 'camice' non serve a nulla, lascialo nell''armadietto'"); 177. p43.setLuogo(5); 178. insPersonaggi->inserisci(43, p43); 180. //luogo 6 51-60 181. p51.setNome("Cid"); 182. p51.setFrasi("Prendi il 'casco' Potrebbe tornarti utile","Non Vedo nulla di utile qui"); 183. p51.setOgg(**true**,50,51); 184. p51.setFrasiconOgg1("Prendi la tuta dalla cabina di pilotaggio","La tuta non ti serve, vai direttametne all'esterno dell'astronave"); 185. p51.setFrasiconOgg2("Visto che hai la tuta, puoi uscire all'esterno dell'astronave","lascia il casco prima di uscire dall'astronave"); 186. p51.setLuogo(6); 187. //modifiche D'Andria Dresda aggiunta dialogo a Cid 188. Dialogo dialogocid=Dialogo("Cid"); 189. dialogocid=caricaDialoghiDaFile("Cid"); 190. p51.setDialogo(&dialogocid); 191. //fine modifiche 192. insPersonaggi->inserisci(51, p51); 193. p52.setNome("Stainer"); 194. p52.setFrasi("Dopo aver preso il 'casco' cerca la 'tuta'","Ripostati un po sul letto, potrebbe servirti a schiarire le idee"); 195. p52.setLuogo(6); 196. insPersonaggi->inserisci(52, p52); 197. p53.setNome("Bruce"); 198. p53.setFrasi("Non uscire all'esterno della nave senza 'casco' e 'tuta'","la 'tuta' e il 'casco' non servo per uscire all'esterno dell'astronave"); 199. p53.setLuogo(6); 200. insPersonaggi->inserisci(53, p53); 202. //luogo 7 61-70 203. p61.setNome("Dom"); 204. p61.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senza 'casco' altrimenti morirai"); 205. p61.setLuogo(7); 206. insPersonaggi->inserisci(61, p61); 207. p62.setNome("Gabriel"); 208. p62.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senza 'casco' altrimenti morirai"); 209. p62.setOgg(**true**,50); 210. p62.setFrasiconOgg1("Hai il il 'casco', ma ti manca la 'tuta' per poter uscire dal'astronave","hai il 'casco', premi rosso così potrai uscire dall'astonave"); 211. p62.setLuogo(7); 212. insPersonaggi->inserisci(62, p62); 213. p63.setNome("Martino"); 214. p63.setFrasi("non 'premere il rosso' senza avere il 'casco'","'premi il rosso' senza 'casco' altrimenti morirai"); 215. p63.setOgg(**true**,50,51); 216. p63.setFrasiconOgg1("Hai il il 'casco', ma ti manca la 'tuta' per poter uscire dal'astronave","hai il 'casco', premi rosso così potrai uscire dall'astonave"); 217. p63.setFrasiconOgg2("Hai sia il 'casco' che la 'tuta' ora puoi uscire dal'astronave","lascia il 'casco', poi premi rosso così potrai uscire dall'astonave"); 218. p63.setLuogo(7); 219. insPersonaggi->inserisci(63, p63); 221. //luogo 8 71-80 222. p71.setNome("Samuel"); 223. p71.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo bastasse fare il contrario?","Il 'manuale' descrive come attivare il reattore, e se per spegnerlo bastasse fare la stessa procedura?"); 224. p71.setLuogo(8); 225. insPersonaggi->inserisci(71, p71); 226. p72.setNome("Shain"); 227. p72.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo bastasse fare il contrario?","Il 'manuale' descrive come attivare il reattore, e se per spegnerlo bastasse fare la stessa procedura?"); 228. p72.setLuogo(8); 229. insPersonaggi->inserisci(72, p72); 230. p73.setNome("Zick"); 231. p73.setFrasi("Il 'manuale' descrive come attivare il reattore, e se per spegnerlo bastasse fare il contrario?","Il 'manuale' descrive come attivare il reattore, e se per spegnerlo bastasse fare la stessa procedura?"); 232. p73.setLuogo(8); 233. insPersonaggi->inserisci(73, p73); 235. //luogo 9 81-90 236. p81.setNome("Lance"); 237. p81.setFrasi("Il 'secondo pilota' e' a nord","si tolga il 'casco' e' sicuro qui"); 238. p81.setLuogo(9); 239. insPersonaggi->inserisci(81, p81); 240. p82.setNome("Lauren"); 241. p82.setFrasi("Il 'secondo pilota' e' a nord","si tolga il 'casco' e' sicuro qui"); 242. p82.setLuogo(9); 243. insPersonaggi->inserisci(82, p82); 244. p83.setNome("Lucas"); 245. p83.setFrasi("Il 'secondo pilota' e' a nord","si tolga il 'casco' e' sicuro qui"); 246. p83.setLuogo(9); 247. insPersonaggi->inserisci(83, p83); 249. //luogo 10 91-100 250. p91.setNome("Andre'"); 251. p91.setFrasi("Soccorra il 'secondo pilota', e nei guai","il 'secondo pilota' e' andato perso nello spazio, e' inutile cercarlo"); 252. p91.setLuogo(10); 253. insPersonaggi->inserisci(91, p91); 254. p91.setNome("George"); 255. p91.setFrasi("Soccorra il 'secondo pilota', e nei guai","il 'secondo pilota' e' andato perso nello spazio, e' inutile cercarlo"); 256. p91.setLuogo(10); 257. insPersonaggi->inserisci(92, p91); 258. p91.setNome("Oscar"); 259. p91.setFrasi("Soccorra il 'secondo pilota', e nei guai","il 'secondo pilota' e' andato perso nello spazio, e' inutile cercarlo"); 260. p91.setLuogo(10); 261. insPersonaggi->inserisci(93, p91); 263. //aggiunta modifiche D'Andria Dresda al progetto di Federica Forte caricamento dialoghi da file 264. scrivistatosufile("Cid",1); 265. scrivistatosufile("Vincent",1); 266. scrivistatosufile("Kail",1); 267. //fine modifiche 268. } |

#### az\_parla

|  |
| --- |
| 1. **void** Gioco::az\_parla(Personaggi pg){ 2. **int** ris\_controllo = 0; 3. parola2[0]=toupper(parola2[0]); 4. **if**(parola2==pg.getNome()){ 5. //modifiche effettuate da D'Andria Dresda per far dialogare i personaggi 6. **if**(pg.getNome()=="Cid" || pg.getNome()=="Vincent" || pg.getNome()=="Kail"){ 7. **int** risposta; 8. **int** statodialogo=caricastatodialogo(pg.getNome()); //leggo da file l'ultimo stato del dialogo 9. pg.getDialogo()->setstato(statodialogo); 10. pg.getDialogo()->visualizzadomande(); 11. **bool** dialogofinito=**false**; 12. **do**{ 13. cout << endl<< "Digita numero risposta -----> "; 14. cin >>risposta; 15. cout << endl; 16. dialogofinito=pg.getDialogo()->rispondi(risposta); 17. **if**(risposta==0){ 18. scrivistatosufile(pg.getNome(),pg.getDialogo()->getstato()); //in caso di risposta 0 si salva lo stato del dialogo, per riprenderlo dallo stesso punto la volta successiva che si incrontrerà lo stesso personaggio 19. } 20. **if** (pg.getDialogo()->isFoglia()){ 21. dialogofinito=**true**; 22. scrivistatosufile(pg.getNome(),1);//nel caso si raggiunga una foglia del dialogo si reimposta lo stato a 1 23. } 24. } 25. **while**(!dialogofinito); 26. } 27. **else**{ 28. **if**(pg.getIfNec()) 29. { 30. ris\_controllo=controlla(); 31. **if**(ris\_controllo==0) 32. pg.SelezionaFrase2(); 33. **else** **if**(ris\_controllo==1) 34. pg.SelezionaFrase1(); 35. **else** 36. parla(pg); 37. } 38. **else** 39. parla(pg); 40. } 41. } 42. **else** 43. cout << parola2 << " non e' in questa stanza"<<endl; 44. } |

### File aggiunti

#### AlberoNario

##### AlberoNario.h:

|  |
| --- |
| 1. #ifndef ALBERONARIO\_H 2. #define ALBERONARIO\_H 3. #include <exception> 4. #include "BinAlbero.h" 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. \*/ 15. **template** <**class** tipoelem> 16. **class** Albero{ 17. **public**: 18. //DEFINIZIONE DI NODO 19. **typedef** **typename** BinAlbero<tipoelem>::Nodo nodo; 20. //DEFINIZIONE DI NODO NULLO 21. **static** constexpr **typename** BinAlbero<tipoelem>::Nodo nil =BinAlbero<tipoelem>::nil; 22. //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), FALSO 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 ALTRIMENTI 43. **bool** ultimofratello(nodo) **const**; 44. //METODO CHE RESTITUISCE IL NODO FRATELLO SUCCESSIVO DEL NODO PASSATO 45. nodo succfratello(nodo)**const**; 46. //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO COME NODO PRIMOFIGLIO DEL NODO PASSATO 47. **void** insprimosottoalbero(nodo, Albero&); 48. //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO COME UN NODO FRATELLO SUCCESSIVO DEL NODO PASSATO 49. **void** inssottoalbero(nodo, Albero&); 50. //METODO CHE PASSATO UN NODO CANCELLA LO STESSO E TUTTI I SUOI DISCENDENTI 51. **void** cancsottoalbero(nodo); 52. //METODO CHE SCRIVE L'ETICHETTA DI TIPO TIPOELEM ALL'INTERNO DEL NODO PASSATO 53. **void** scrivinodo(nodo&, tipoelem); 54. //METODO CHE RESTITUISCE L'ETICHETTA DEL NODO PASSATO 55. tipoelem legginodo(nodo) **const**; 57. **private**: 58. BinAlbero<tipoelem> tree; 59. //METODO CHE COPIA UN ALBERO PASSATO PARTENDO DAL NODO SINISTRO 60. **void** copia\_albero\_sx(**const** BinAlbero<tipoelem>& ,**const** nodo& ,nodo); 61. //METODO CHE COPIA L'ALBERO PASSATO PARTENDO DAL NODO DESTRO 62. **void** copia\_albero\_dx(**const** BinAlbero<tipoelem>& ,**const** nodo& ,nodo); 63. };  66. **template** <**class** tipoelem> 67. Albero<tipoelem>::Albero(){ 68. creaalbero(); 69. } 71. **template** <**class** tipoelem> 72. Albero<tipoelem>::Albero(**const** Albero& a){ 73. **if**(!a.tree.binalberovuoto()) 74. { 75. tree.insbinradice(); 76. nodo n = tree.binradice(); 77. nodo secondo\_albero = a.tree.binradice(); 78. tree.scrivinodo(n, a.tree.legginodo(secondo\_albero)); 79. **if**(a.tree.sinistrovuoto(a.tree.binradice())) 80. { 81. **if**(!a.tree.destrovuoto(a.tree.binradice())) 82. copia\_albero\_dx(a.tree,a.tree.figliodestro(a.tree.binradice()),tree.binradice()); 83. } 84. **else** 85. copia\_albero\_sx(a.tree,a.tree.figliosinistro(a.tree.binradice()),tree.binradice()); 86. } 87. } 89. **template** <**class** tipoelem> 90. Albero<tipoelem>::~Albero(){ 91. //RICHIAMA AUTOMATICAMENTE IL DISTRUTTORE 92. //DELL'ALBERO BINARIO 93. } 95. **template** <**class** tipoelem> 96. **void** Albero<tipoelem>::creaalbero(){ 97. //RICHIAMA AUTOMATICAMENTE IL COSTRUTTORE 98. //DELL'ALBERO BINARIO 99. }  102. **template** <**class** tipoelem> 103. **bool** Albero<tipoelem>::alberovuoto() **const**{ 104. **return** tree.binalberovuoto(); 105. } 107. **template** <**class** tipoelem> 108. **void** Albero<tipoelem>::insradice(){ 109. tree.insbinradice(); 110. } 112. **template** <**class** tipoelem> 113. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::radice() **const**{ 114. **return** tree.binradice(); 115. } 117. **template** <**class** tipoelem> 118. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::padre(**typename** Albero<tipoelem>::nodo n){ 119. **return** tree.binpadre(n); 120. } 122. **template** <**class** tipoelem> 123. **bool** Albero<tipoelem>::foglia(**typename** Albero<tipoelem>::nodo n) **const**{ 124. **if**(tree.sinistrovuoto(n)) 125. **return** **true**; 126. **else** 127. **return** **false**; 128. } 130. **template** <**class** tipoelem> 131. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::primofiglio(**typename** Albero<tipoelem>::nodo n) **const**{ 132. **return** tree.figliosinistro(n); 133. } 135. **template** <**class** tipoelem> 136. **bool** Albero<tipoelem>::ultimofratello(**typename** Albero<tipoelem>::nodo n) **const**{ 137. **if**(tree.destrovuoto(n)) 138. **return** **true**; 139. **else** 140. **return** **false**; 141. } 143. **template** <**class** tipoelem> 144. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::succfratello(**typename** Albero<tipoelem>::nodo n)**const**{ 145. **return** tree.figliodestro(n); 146. } 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)); 155. **if**(!other.sinistrovuoto(radice\_sottoalbero)) 156. copia\_albero\_sx(other, other.figliosinistro(radice\_sottoalbero), sx); 157. **if**(!other.destrovuoto(radice\_sottoalbero)) 158. copia\_albero\_dx(other, other.figliodestro(radice\_sottoalbero), sx); 159. } 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)); 168. **if**(!other.sinistrovuoto(radice\_sottoalbero)) 169. copia\_albero\_sx(other, other.figliosinistro(radice\_sottoalbero), dx); 170. **if**(!other.destrovuoto(radice\_sottoalbero)) 171. copia\_albero\_dx(other, other.figliodestro(radice\_sottoalbero), dx); 172. } 174. **template** <**class** tipoelem> 175. **void** Albero<tipoelem>::insprimosottoalbero(**typename** Albero<tipoelem>::nodo n, Albero& a){ 176. nodo temp; 177. **if**(tree.sinistrovuoto(n)){ 178. copia\_albero\_sx(a.tree,a.tree.binradice(),n); 179. } 180. **else**{ 181. Albero<tipoelem> T; 182. T.insradice(); 183. temp=T.radice(); 184. T.scrivinodo(temp,tree.legginodo(tree.figliosinistro(n))); 186. **if**(tree.sinistrovuoto(tree.figliosinistro(n))){ 187. **if**(!tree.destrovuoto(tree.figliosinistro(n))){ 188. copia\_albero\_dx(tree,tree.figliodestro(tree.figliosinistro(n)),temp); 189. } 190. } 191. **else**{ 192. copia\_albero\_sx(tree,tree.figliosinistro(tree.figliosinistro(n)),temp); 193. } 195. tree.cancsottobinalbero(tree.figliosinistro(n)); 197. copia\_albero\_sx(a.tree,a.tree.binradice(),n); 198. temp=tree.figliosinistro(n); 199. copia\_albero\_dx(T.tree,T.radice(),temp); 200. } 201. }  204. **template** <**class** tipoelem> 205. **void** Albero<tipoelem>::inssottoalbero(**typename** Albero<tipoelem>::nodo n, Albero& a){ 206. /\* L'albero è ottenuto aggiungendo il sottoalbero a di radice r dove r diventa il fratello successivo 207. di n. n non è la radice\*/ 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(); 218. temp=T.radice(); 219. T.scrivinodo(temp,tree.legginodo(tree.figliodestro(n))); 221. **if**(tree.sinistrovuoto(tree.figliodestro(n))){ 222. **if**(!tree.destrovuoto(tree.figliodestro(n))){ 223. copia\_albero\_dx(tree,tree.figliodestro(tree.figliodestro(n)),temp); 224. } 225. } 226. **else**{ 227. copia\_albero\_sx(tree,tree.figliosinistro(tree.figliodestro(n)),temp); 228. } 230. cancsottoalbero(tree.figliodestro(n)); 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. } 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. } 245. **template** <**class** tipoelem> 246. **void** Albero<tipoelem>::scrivinodo(**typename** Albero<tipoelem>::nodo& n, tipoelem elem){ 247. tree.scrivinodo(n,elem); 248. } 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 |

##### BinAlbero.h

|  |
| --- |
| 1. #ifndef BINALBERI\_H\_INCLUDED 2. #define BINALBERI\_H\_INCLUDED 3. #include "Nodo\_Albero\_Binario.h" 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{ 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, 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 figliodestro(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** cancsottobinalbero(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** insfigliodestro(Nodo); 55. **private**: 56. Nodo radice; 57. };  60. **template** <**class** TIPOETICHETTA> 61. BinAlbero<TIPOETICHETTA>::BinAlbero(){ 62. creabinalbero(); 63. } 65. **template** <**class** TIPOETICHETTA> 66. BinAlbero<TIPOETICHETTA>::~BinAlbero(){ 67. /\* if(radice!=nil) 68. cancsottobinalbero(binradice()); 69. \*/ 71. } 73. **template** <**class** TIPOETICHETTA> 74. **void** BinAlbero<TIPOETICHETTA>::creabinalbero(){ 75. radice=nil; 76. } 78. **template** <**class** TIPOETICHETTA> 79. **bool** BinAlbero<TIPOETICHETTA>::binalberovuoto()**const**{ 80. **if**(radice==nil) 81. **return** **true**; 82. **else** 83. **return** **false**; 84. } 86. **template** <**class** TIPOETICHETTA> 87. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binradice()**const**{ 88. **return** radice; 89. } 91. **template** <**class** TIPOETICHETTA> 92. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binpadre(Nodo padre)**const**{ 93. **return** padre->getPadre(); 94. } 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. } 104. **template** <**class** TIPOETICHETTA> 105. **bool** BinAlbero<TIPOETICHETTA>::destrovuoto(Nodo dx)**const**{ 106. **if**(dx->getFiglioDx()==nil) 107. **return** **true**; 108. **else** 109. **return** **false**; 110. } 112. **template** <**class** TIPOETICHETTA> 113. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliosinistro(Nodo sx)**const**{ 114. **return** sx->getFiglioSx(); 115. } 117. **template** <**class** TIPOETICHETTA> 118. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliodestro(Nodo dx)**const**{ 119. **return** dx->getFiglioDx(); 120. } 122. **template** <**class** TIPOETICHETTA> 123. **void** BinAlbero<TIPOETICHETTA>::costrbinalbero(BinAlbero<TIPOETICHETTA> &A,BinAlbero<TIPOETICHETTA> &B){ 124. radice=**new** Nodo; 125. radice->setPadre(nil); 127. **if**(!A.binalberovuoto()){ 128. radice->setFiglioSx(A.binradice()); 129. radice->getFiglioSx()->setPadre(radice); 130. } 131. **else** 132. radice->setFiglioSx(nil);  135. **if**(!B.binalberovuoto()){ 136. radice->setFiglioDx(B.binradice()); 137. radice->getFiglioDx()->setPadre(radice); 138. } 139. **else** 140. radice->setFiglioDx(nil); 141. } 143. **template** <**class** TIPOETICHETTA> 144. **void** BinAlbero<TIPOETICHETTA>::cancsottobinalbero(Nodo r){ 146. **if**(!sinistrovuoto(r)) 147. cancsottobinalbero(r->getFiglioSx()); 149. **if**(!destrovuoto(r)) 150. cancsottobinalbero(r->getFiglioDx()); 152. **if**(radice!=r) 153. { 154. Nodo temp; 155. temp=binpadre(r); 157. **if**(temp->getFiglioSx()==r) 158. temp->setFiglioSx(nil); 159. **else** 160. temp->setFiglioDx(nil); 161. } 162. **else** 163. radice=nil; 165. **delete** r; 167. } 169. **template** <**class** TIPOETICHETTA> 170. TIPOETICHETTA BinAlbero<TIPOETICHETTA>::legginodo(Nodo n)**const**{ 171. **return** n->getEtichetta(); 172. } 174. **template** <**class** TIPOETICHETTA> 175. **void** BinAlbero<TIPOETICHETTA>::scrivinodo(Nodo n, TIPOETICHETTA e){ 176. n->setEtichetta(e); 177. } 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. } 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. } 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. } 205. #endif // BINALBERI\_H\_INCLUDED |

##### Nodo\_Albero\_Binario.h

|  |
| --- |
| 1. #ifndef NODO\_ALBERO\_BINARIO\_H\_INCLUDED 2. #define NODO\_ALBERO\_BINARIO\_H\_INCLUDED 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. };  47. **template** <**class** TIPOETICHETTA> 48. Cella\_Binalbero<TIPOETICHETTA>::Cella\_Binalbero(){ 49. FiglioDx=nullptr; 50. FiglioSx=nullptr; 51. Padre=nullptr; 52. } 54. **template** <**class** TIPOETICHETTA> 55. Cella\_Binalbero<TIPOETICHETTA>::~Cella\_Binalbero(){ 56. } 58. **template** <**class** TIPOETICHETTA> 59. **void** Cella\_Binalbero<TIPOETICHETTA>::setFiglioSx(Cella\_Binalbero<TIPOETICHETTA>\* sx){ 60. FiglioSx=sx; 61. } 63. **template** <**class** TIPOETICHETTA> 64. **void** Cella\_Binalbero<TIPOETICHETTA>::setFiglioDx(Cella\_Binalbero<TIPOETICHETTA>\* dx){ 65. FiglioDx=dx; 66. } 68. **template** <**class** TIPOETICHETTA> 69. **void** Cella\_Binalbero<TIPOETICHETTA>::setPadre(Cella\_Binalbero<TIPOETICHETTA>\* p){ 70. Padre=p; 71. } 73. **template** <**class** TIPOETICHETTA> 74. **void** Cella\_Binalbero<TIPOETICHETTA>::setEtichetta(TIPOETICHETTA& e){ 75. etichetta=e; 76. } 78. **template** <**class** TIPOETICHETTA> 79. Cella\_Binalbero<TIPOETICHETTA>\* Cella\_Binalbero<TIPOETICHETTA>::getFiglioSx(){ 80. **return** FiglioSx; 81. } 83. **template** <**class** TIPOETICHETTA> 84. Cella\_Binalbero<TIPOETICHETTA>\* Cella\_Binalbero<TIPOETICHETTA>::getFiglioDx(){ 85. **return** FiglioDx; 86. } 88. **template** <**class** TIPOETICHETTA> 89. Cella\_Binalbero<TIPOETICHETTA>\* Cella\_Binalbero<TIPOETICHETTA>::getPadre(){ 90. **return** Padre; 91. } 93. **template** <**class** TIPOETICHETTA> 94. TIPOETICHETTA Cella\_Binalbero<TIPOETICHETTA>::getEtichetta(){ 95. **return** etichetta; 96. } 98. **template** <**class** TIPOETICHETTA> 99. **bool** Cella\_Binalbero<TIPOETICHETTA>::operator == (Cella\_Binalbero<TIPOETICHETTA>  b){ 100. **if**(getEtichetta()==b.getEtichetta()) 101. **return** **true**; 102. **else** 103. **return** **false**; 104. } 105. **template** <**class** TIPOETICHETTA> 106. Cella\_Binalbero<TIPOETICHETTA>::Cella\_Binalbero(**const** Cella\_Binalbero& c){ 107. etichetta=c.getEtichetta(); 108. FiglioDx=c.getFiglioDx(); 109. FiglioSx=c.getFiglioSx(); 110. Padre=c.getPadre(); 111. } 113. **template** <**class** TIPOETICHETTA> 114. **void** Cella\_Binalbero<TIPOETICHETTA>::operator=(**const** Cella\_Binalbero& c ){ 115. etichetta=c.getEtichetta(); 116. Padre=c.getPadre(); 117. FiglioDx=c.getFiglioDx(); 118. FiglioSx=c.getFiglioSx(); 119. } 120. #endif // NODO\_ALBERO\_BINARIO\_H\_INCLUDED |

#### Domanda

##### Domanda.h:

|  |
| --- |
| 1. #ifndef DOMANDA\_H 2. #define DOMANDA\_H 3. #include "string" 4. **using** **namespace** std; 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 15. **private**: 16. string domanda; 17. **int** codicedomanda; 18. };   22. #endif // DOMANDA\_H |

##### Domanda.cpp:

|  |
| --- |
| 1. #include "Domanda.h" 3. Domanda::Domanda() {} 5. //costruttore di copia 6. Domanda::Domanda(**const** Domanda& d){ 7. domanda = d.domanda; 8. codicedomanda = d.codicedomanda; 9. } 11. Domanda::~Domanda() {} 13. Domanda::Domanda(**int** codice, string stringa){ 14. **this**->codicedomanda=codice; 15. **this**->domanda=stringa; 16. } 18. **int** Domanda::leggicodice(){ 19. **return** **this**->codicedomanda; 20. } 22. string Domanda::leggidomanda(){ 23. **return** **this**->domanda; 24. } |

#### Dialogo

##### 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**); 18. string leggiPersonaggio(); 19. **int** getstato(); 20. **void** setstato(**int**); 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** inserisciopzionesuccessiva(Albero<Domanda>::nodo,**int**, string); 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. **bool** isFoglia(); 39. **private**: 40. string personaggio;//nome personaggio a cui è associato il dialogo 41. **int** stato;//stato attuale del dialogo 42. Albero<Domanda> dialogo;//albero contenente le domande del dialogo 43. }; 45. #endif // DIALOGO\_H |

##### Dialogo.cpp:

|  |
| --- |
| 1. #include "Dialogo.h" 3. Dialogo::Dialogo() {} 4. Dialogo::~Dialogo() { 5. } 7. Dialogo::Dialogo(**const** Dialogo& d){ 8. personaggio = d.personaggio; 9. stato = d.stato; 10. dialogo = Albero<Domanda>(d.dialogo); 11. } 13. Dialogo::Dialogo(string nomepersonaggio){ 14. **this**->personaggio=nomepersonaggio; 15. **this**->stato=1; 16. } 18. **void** Dialogo::setstato(**int** numero){ 19. **this**->stato=numero; 20. } 22. **int** Dialogo::getstato(){ 23. **return** (**this**->stato); 24. } 25. string Dialogo::leggiPersonaggio(){ 26. **return** (**this**->personaggio); 27. } 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. inserisciopzionesuccessiva(indicenodo,codice,frase); 46. } 47. } 48. } 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; 54. **if**(codice==dialogo.legginodo(indice).leggicodice()) 55. nododaritornare=indice; 57. **else**{ 58. **if**(!dialogo.foglia(indice)){ 59. nodo=dialogo.primofiglio(indice); 60. **while**(!dialogo.ultimofratello(nodo) && nododaritornare==Albero<Domanda>::nil){ 61. nododaritornare=trovanodo(nodo,codice); 62. nodo=dialogo.succfratello(nodo); 63. } 64. **if** (nododaritornare==Albero<Domanda>::nil) 65. nododaritornare=trovanodo(nodo,codice); 66. } 67. } 68. **return** nododaritornare; 69. }  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. } 80. **void** Dialogo::inserisciprimaopzione(Albero<Domanda>::nodo domandacorrente,**int** codice, string frase){ 81. Albero<Domanda> alberoopzione; 82. Albero<Domanda>::nodo nodoopzione; 83. Domanda nuovadomanda= Domanda(codice,frase); 84. alberoopzione.insradice(); 85. nodoopzione=alberoopzione.radice(); 86. alberoopzione.scrivinodo(nodoopzione, nuovadomanda); 87. dialogo.insprimosottoalbero(domandacorrente,alberoopzione); 88. } 90. **void** Dialogo::inserisciopzionesuccessiva(Albero<Domanda>::nodo nodoprecedente,**int** codice, string frase){ 91. Albero<Domanda> alberoopzione; 92. Albero<Domanda>::nodo nodoopzione; 93. Domanda nuovadomanda= Domanda(codice,frase); 94. alberoopzione.insradice(); 95. nodoopzione=alberoopzione.radice(); 96. alberoopzione.scrivinodo(nodoopzione, nuovadomanda); 97. dialogo.inssottoalbero(nodoprecedente,alberoopzione); 98. } 100. **void** Dialogo::visualizzadomande(){ 101. **int** domandadavisualizzare=**this**->stato; 103. Albero<Domanda>::nodo indicenodo=Albero<Domanda>::nil; 104. indicenodo = trovanodo(dialogo.radice(),domandadavisualizzare); 105. cout<<dialogo.legginodo(indicenodo).leggidomanda(); 106. } 108. **void** Dialogo::cambiastato(**int** risposta){ 109. **int** statosuccessivo=(**this**->stato)\*10+risposta; 110. Albero<Domanda>::nodo nodosuccessivo=Albero<Domanda>::nil; 111. nodosuccessivo = trovanodo(dialogo.radice(),statosuccessivo); 112. **if**(nodosuccessivo!=Albero<Domanda>::nil) 113. **this**->stato=statosuccessivo; 114. **else** 115. cout<< "Risposta non valida\n"; 116. visualizzadomande(); 117. } 119. **bool** Dialogo::rispondi(**int** risposta){ 120. **if** (risposta==0) 121. **return** **true**; 122. **else**{ 123. cambiastato(risposta); 124. **return** **false**; 125. } 126. } 128. **bool** Dialogo::dialogoVuoto(){ 129. **return**(dialogo.alberovuoto()); 130. } 132. **bool** Dialogo::isFoglia(){ 133. **int** domandadavisualizzare=**this**->stato; 134. Albero<Domanda>::nodo indicenodo=Albero<Domanda>::nil; 135. indicenodo = trovanodo(dialogo.radice(),domandadavisualizzare); 136. **return**(dialogo.foglia(indicenodo)); 137. } |

##### Dialoghi.txt

|  |
| --- |
| 1. -Cid|1\*Ciao, sono Cid!!$$Credi in Dio?$1:si$2:no$0:interrompi dialogo$# 2. -Cid|11\*Benissimo, sei gia' andato in chiesa?$$1:Si ci sono stato$2:Non ancora$0:interrompi dialogo$# 3. -Cid|12\*Male, molto male. Solo Lui puo' aiutarti!$# 4. -Cid|111\*Bene, ti sei confessato??$$1:Si$2:no$0:interrompi dialogo$# 5. -Cid|112\*Vai finche' sei tempo e purifica la tua anima confessandoti!$# 6. -Cid|1111\*Bravissimo!! Dio ti condurra' verso la salvezza!$# 7. -Cid|1112\*Tornaci e confessati! Che Dio ti benedica!!$# 8. -Vincent|1\*Ciao, sono Vincent!$$Preferisci il fast food oppure il ristorante?$1:Fast food$2:Ristorante$0:Interrompi dialogo$# 9. -Vincent|11\*Interessante, ci sei gia' stato?$1:Si$2:Non ancora$0:Interrompi dialogo$# 10. -Vincent|12\*Benissimo, ti consiglio di andarci per gustare delle specialita' galattiche!$Cosa ti piace mangiare?$1:Carne$2:Pesce$3:Verdure$0:Interrompi dialogo$# 11. -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 li di passaggio e non ho mangiato nulla$0:Interrompi dialogo$# 12. -Vincent|112\*Cosa aspetti?! Vai, prima che tu muoia di fame!!!$# 13. -Vincent|121\*Ottimo, credo che il loro arrosto sia uno dei piu' buoni in tutto l'universo!$# 14. -Vincent|122\*Hai gia' provato il loro misto di frutti di mare?$1:Si$2:No$0:interrompi dialogo$# 15. -Vincent|123\*Prova gli spinaci alla panna, sono deliziosi$# 16. -Vincent|1111\*Gia', lo penso anche io, prova anche il ristorante, l'ho trovato adorabile.$# 17. -Vincent|1112\*Mi dispiace, vorra' dire che andrai al ristorante per mangiare qualcosa.$# 18. -Vincent|1113\*Male, prova il loro gelato alla panna e' delizioso, ricordati di usare il coupon!$# 19. -Vincent|1221\*Io l'ho apprezzato molto!$# 20. -Vincent|1222\*Provalo, non te ne pentirai!$# 21. -Kail|1\*Ciao, sono Kail!$$Hai bisogno di cure?$1:Si, grazie$2:No, grazie$0:interrompi dialogo$# 22. -Kail|11\*Spero nulla di grave, hai gia' preso la tessera sanitaria?$1:Si$2:No$0:interrompi dialogo$# 23. -Kail|12\*Apprezzo il fatto che tu stia bene, se dovesse servirti recati al pronto soccorso.$# 24. -Kail|111\*Benissimo, corri al pronto soccorso e usa il terminale!$# 25. -Kail|112\*Cosa aspetti?! Hai almeno idea di dove si trovi?$1:Si, so dov'e'$2:Non ne ho la piu' pallida idea$0:interrompi dialogo$# 26. -Kail|1121\*Ottimo, corri a prenderla e recati al pronto soccorso!!!$# 27. -Kail|1122\*Come hai fatto a non vederla?! E' nella tua cabina! Prendila e vai al pronto soccorso!!!$# 28. ? |

##### Statidialoghi.txt

|  |
| --- |
| 1. ? |

## Sala scommesse e simulatore

### 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; //Strina 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 nbumero 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 si 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 -,

### 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 orgininaria 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; //inizzializzazione 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 de 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) //sovraccarico 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

## Auditorium

### Gioco.h

Inserite le dichiarazioni delle variabili, delle strutture dati necessarie per il funzionamento del Jukebox e del pannello delle luci e le funzioni di salvataggio e ripristino del Jukebox e delle luci.

Void salvaJukeBoxLuci(); //salva lo stato del juke e delle luci  
 void caricaJukeBoxLuci() //ripristina lo stato del juke e delle luci

Canzone canzone; Inizializzazione classe canzone  
 Lista<Canzone> canzoni; Inizializzazione della lista di canzoni del Jukebox  
 Lista<Canzone>::posizione canzoneScelta;   
 bool jukeboxAttivo; //Verifica se il jukebox è usato  
 int canzoneInRiproduzione; //Serve per il salvataggio  
 Coda<Canzone> playlist; //Salva la playlist dei brani ascoltati

Luce luce;  
 Insieme<Luce> luciAuditoriumAccese; - permette di vedere quali sono le luci accese nell'auditorium  
 string nome\_luci\_auditorium[6] = {"fittizio","sulla destra, avanti","sulla sinistra, avanti","sulla destra, dietro","sulla sinistra, dietro","sul palco"}; - array di stringhe dei nomi delle luci

### 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

### 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

#### Accendi il proiettore rotto

void Astro::azione\_105(){

interfaccia.scrivi("e' un proiettore non funzionante, la lente e' rotta");

}

#### Guarda lo schermo nell'auditorium

void Astro::azione\_106(){

interfaccia.scrivi("Lo schermo non mostra nulla poiche' il proiettore non e' acceso.");

}

#### 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...");

}

#### Parla al microfono

void Astro::azione\_108(){

interfaccia.scrivi("La mia voce rieccheggia in tutto l'Auditorium! Meglio smetterla prima di ricevere una lavata di capo...");

}

#### Interagisci con il Jukebox Antico

void Astro::azione\_109(){

string decisione; //per contenere la decisione dell'utente

interfaccia.scrivi("È un antico Jukebox! L'acustica qua e' perfetta!");

jukeboxAttivo = true;

while(jukeboxAttivo){

canzone = canzoni.leggilista(canzoneScelta);

playlist.incoda(canzone); //inserisce nella coda la canzone in riproduzione.

cout << "\n";

cout << canzone; //possibile grazie all'overload dell'operatore

interfaccia.scrivi("Posso andare alla prima canzone, a quella precedente o a quella successiva. Cosa faccio?");

cin >> decisione;

if ( (decisione.compare("prima") == 0) || (decisione.compare("Prima") == 0) || (decisione.compare("PRIMA") == 0) ){   
 canzoneScelta = canzoni.primolista();   
 canzoneInRiproduzione = 0 }  
   
 else if( (decisione.compare("successiva") == 0) || (decisione.compare("Successiva") == 0) || (decisione.compare("SUCCESSIVA") == 0) ){

if ( canzoni.finelista(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.preclista(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 } }

#### Guarda gli strumenti Musicali

void Astro::azione\_110(){

interfaccia.scrivi("Sul palco dell'Auditorium sono presenti una chitarra, un violino e un pianoforte"); }

#### 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...");}

}

#### 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" ; }

}

}

#### 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;

### 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 accesse**

**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(); }

## 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

### 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,**33**2414000200,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*

### 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

### Astro.cpp

Nel file astro.cpp sono state apportate le seguenti modifiche:

#### Inserimento vocaboli

//INIZIO modifiche SALVATORE VESTITA

vocabolario.inserisci("siediti", **94**);

//FINE modifiche SALVATORE VESTITA

#### 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

#### 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

#### 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 sottrato 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 DESIDERA 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\n**HAI 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("#################################################################################");

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

### 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

### 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

### 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;

}

## 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

### 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
* È stata modificata la linea di codice relativa alla cabina del secondo pilota, che permette ora, scendendo di raggiungere la palestra

5,Nella cabina del secondo pilota,001802171634,via 2,2,2,via 7,2,2

### 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

### Astro.cpp

Nel fine Astro.cpp sono state apportate le seguenti modifiche:

#### Inserimento vocaboli

//INIZIO modifiche VINCENZO GIANNUZZO

vocabolario.inserisci("panca", 57);

vocabolario.inserisci("tapis", 47);

vocabolario.inserisci("roulant", 47);

vocabolario.inserisci("scheda", 84);

vocabolario.inserisci("schede", 84);

//FINE modifiche VINCENZO GIANNUZZO

#### Inserimento azioni

//INIZIO modifiche VINCENZO GIANNUZZO

azioni.inserisci(342957, 118); //usa panca

azioni.inserisci(342947, 119); //usa tapis

azioni.inserisci(342999, 120); //usa terminale

azioni.inserisci(341099, 120); //guarda terminale

azioni.inserisci(342984, 121); //usa schede

//FINE modifiche VINCENZO GIANNUZZO

#### 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

#### 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 rimanente per finire il gioco.

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);

}

void Astro::azione\_119() // usa tapis

{

interfaccia.scrivi("Per quanti minuti vuoi correre ?");

int rip;

string risp;

getline(cin, risp);

stringstream(risp) >> rip;

Palestra::usaTapis(sFisico, rip);

}

L’azione 120 invoca la funzione di usare il terminale, contenuta sempre all’interno della classe “Palestra”.

void Astro::azione\_120() //usa/guarda terminale

{

Palestra::usaTerminale();

}

L’azione 121, chiama la funzione di accedere alle schede, inglobata all’interno della classe “Palestra”.

void Astro::azione\_121()

{

Palestra::usaSchede(sFisico);

}

### Astro.h

//INIZIO modifiche VINCENZO GIANNUZZO

void azione\_118(); //Usa la panca

void azione\_119(); //Usa tapis

void azione\_120(); //Leggi terminale informativo

void azione\_121(); //Usa schede

//FINE modifiche VINCENZO GIANNUZZO

### Gioco.cpp

Sono state inserite alcune modifiche alle procedure aggiorna\_tempo(), Le modifiche erano necessarie affinché i valori di costituzione e resistenza di StatoFisico potessero influenzare rispettivamente il tempo perso e le probabilità essere feriti.

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 (sazio <= 120 && sazio >= 80)

{

tempo\_perso+= 1; //Modifica GIANNUZZO

if (sazio <= 90 && sazio >= 80)

interfaccia.scrivi("\nInizio a sentire un leggero languo-rino...");

}

else if (sazio <= 79 && sazio >= 50)

{

tempo\_perso+= 2; //Modifica GIANNUZZO

if (sazio <= 60 && sazio >= 50)

interfaccia.scrivi("\nE' tutto il giorno che non mangio, do-vrei fare una pausa...");

}

if (Salute.GetStatoSalute() <= 40)

{

tempo -= Salute.GetPenalita();

}

else //La penalità non viene ridotta se la salute è troppo bassa

{

tempo\_perso += Salute.GetPenalita();

}

tempo -= sFisico.riduciTempo(tempo\_perso);

//Qui la penalità di tempo totale viene ridotta in base alla costituzione attuale del personaggio

cout << "Tempo residuo: " << tempo << endl;

Nella funzione “esegui”, la probabilità di ricevere ferite viene ridotta dalla resistenza del personaggio

if(!sFisico.feritaEvitata(rand()))//Modifica GIANNUZZO

{

Aggiorna\_Ferite(); //Generatore Ferite

}

### 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;

### StatoFisico.h

Classe che implementa le caratteristiche fisiche del personaggio

class StatoFisico

{

public:

StatoFisico();

~StatoFisico();

inline float getCostituzione() { return costituzione; }

inline void setCostituzione(float cos) { costituzione = cos; }

inline float getResistenza() { return resistenza; }

inline void setResistenza(float res) { resistenza = res; }

bool feritaEvitata(int random);

float riduciTempo(float time);

const static int MAX\_COS;

const static int MAX\_RES;

private:

float costituzione;

float resistenza;

### StatoFisico.cpp

const int StatoFisico::MAX\_COS = 50;

const int StatoFisico::MAX\_RES = 50;

StatoFisico::StatoFisico()

{

costituzione = 0;

resistenza = 0;}

StatoFisico::~StatoFisico()

{

}

float StatoFisico::riduciTempo(float time)

{

if (resistenza == MAX\_RES && costituzione == MAX\_COS)

{

time -= (time / 100) \* 75; }

else

{

time -= (time / 100) \* costituzione; }

return time;

}

bool StatoFisico::feritaEvitata(int random)

{

bool fer = false;

random = (random % 99) + 1;

if (resistenza == MAX\_RES && costituzione == MAX\_COS)

{

fer = (random <= 75); }

else

{

fer = (random <= resistenza); }

return fer;

}

### 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

### Scheda.h

Questa classe rappresenta le singole schede di allenamento. Ogni scheda consiste di un attributo “nome” di tipo stringa e uno chiamato “esercizi” il cui tipo è una Coda<Esercizio> dove <Esercizio> è un tipo struct pubblico definito in Scheda.h e formato dai campi string tipo (ovvero “panca” o “tapis”) e int ripetizioni.

L’utilizzo della coda per rappresentare la lista di esercizi è dovuto alla necessità di mantenere l’ordine in essi verranno eseguiti al momento dell’utilizzo della scheda; tuttavia, poiché il processo di lettura di una coda è distruttivo, si è rivelato necessario implementare un costruttore di copia nel file Coda.h. In questo modo il metodo Scheda::getEsercizi() può restituire una copia dell’intera coda preservando l’originale.

Per altri dettagli riguardanti l’implementazione si rimanda al file Scheda.cpp.

class Scheda

{

public:

Scheda();

Scheda(string);

~Scheda();

struct Esercizio

{

string tipo;

int ripetizioni;

};

string getNome();

void setNome(string);

void addEsercizio(string tipo, int ripetizioni);

Coda<Esercizio> getEsercizi();

private:

string nome;

Coda<Esercizio> esercizi;

## Sala giochi

### 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

### 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);

### 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;

### 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);

}

### 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;

### Metodo load() e save()

Tenendo conto del denaro presente nella slot-machine, il metodo load() del file gioco.cpp è stato modificato come di seguito:

int slot;

svuotaInsieme(slotmachine);

file >> numEuro;

for(int j=1; j<=numEuro; j++)

{

file >> slot;

slotmachine.inserisci(slot);

}

Stessa cosa per il metodo save():

for(int k=1; k<=31; k++)

if(slotmachine.appartiene(k))

numEuro++;

file << numEuro << '\n';

for(int j=1; j<=31; j++)

if(slotmachine.appartiene(j))

file << j << '\n';

Gianluca Scatigna

## Implementazione Biblioteca

### 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, 000000000028, 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, 000000000028, 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

### 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.

### 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

### 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));

### 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();

#### 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 e' 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. }

#### apri vetrina

1. //astro.cpp
2. void Astro::azione\_127() {
3. interfaccia.scrivi("la vetrina e' aperta!\n");
4. luogo\_attuale = 38;
5. }

#### chiudi vetrina

1. //astro.cpp
2. void Astro::azione\_128() {
3. interfaccia.scrivi("la vetrina e' chiusa!\n");
4. luogo\_attuale = 37;
5. }

#### 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

##### 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 costruito   |";
8. cout << "\n| stellare) e' un veicolo spaziale         | un'autentica astronave, molti     |";
9. cout << "\n| progettato per il viaggio interstellare  | scienziati (Freeman Dyson e il    |";
10. cout << "\n| cioe' in grado di raggiungere sistemi    | Progetto Orione) hanno discusso   |";
11. 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. cout << "\n| della space opera) abbonda di storie che | intraprendere in futuro dei viaggi|";
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 hanno |";
32. cout << "\n| hanno partecipato come equipaggi         | specifico ruolo ma sono inseriti  |";
33. cout << "\n| divisi in ordine                         | specialisti di carico per ragioni |";
34. cout << "\n| cronologico rispetto alla missione.      | di spazio.                        |";
35. cout << "\n| Abbreviazioni usate:                     | STS-61-A e' l'unico volo, lanciato|";
36. cout << "\n| PC = Comandante del carico utile         | nel 1985, che ha portato a bordo  |";
37. cout << "\n| MSE = Ingegnere di volo dell'USAF        | equipaggio maggiore di sette      |";
38. cout << "\n| Mir = Lanciato come equipaggio della     | astronauti.                       |";
39. cout << "\n| stazione spaziale Mir                    |                                   |";
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. system("pause");
46. break;
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;

### Implementazione metodi per funzionamento biblioteca

#### 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.succlista(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. }

#### 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) {                        oggetti.set\_luogo(og, -99);                        interfaccia.scrivi("Si e' perso nello spazio.");                        stringa\_risposta = "si e' perso nello spazio!!";    //Modifica PMF(storia)                        storia\_gioco.insStoria(stringa\_comando , stringa\_risposta); //Modifica PMF(storia)                    } else                        lasciato = false;                    return lasciato;\*/ //modifica Gallone
3. bool lasciato = true; //Inizio Modifica Gallone - aggiorno il metodo per integrare le nuove funzionalità
4. riferimento\_tasca = tasca.primolista();
5. riferimento\_zaino = zaino\_frigo.primolista();
6. if (luogo\_attuale == 9 || luogo\_attuale == 10 || luogo\_attuale == 11 || (luogo\_attuale == 7 && parete\_stagna\_aperta == 1)) {
7. if (og >= 35 && og <= 37) {
8. while (!tasca.finelista(riferimento\_tasca)) {
9. if (oggetti.get\_oggetto(tasca.leggilista(riferimento\_tasca)).get\_codice() == oggetti.get\_oggetto(og).get\_codice()) tasca.canclista(riferimento\_tasca);
10. else riferimento\_tasca = tasca.succlista(riferimento\_tasca);
11. }
12. } else if (og >= 25 && og <= 34) {
13. while (!zaino\_frigo.finelista(riferimento\_zaino)) {
14. if (oggetti.get\_oggetto(zaino\_frigo.leggilista(riferimento\_zaino)).get\_codice() == oggetti.get\_oggetto(og).get\_codice()) zaino\_frigo.canclista(riferimento\_zaino);
15. else riferimento\_zaino = zaino\_frigo.succlista(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 SUGLI OGGETTI CHE SI PERDONO
21. if (luogo\_attuale == 38) {
22. int codOggetto = oggetti.get\_oggetto(og).get\_codice();
23. if (codOggetto >= 90) // SE OGGETTO È UN LIBRO
24. {
25. Lista < int > ::posizione p = libriInPrestito.primolista();
26. for ( **int** j = 0;
27. ((!libriInPrestito.finelista(p)) && (p < MAX\_LIBRI)); p = libriInPrestito.succlista(p)) {
28. if (libriInPrestito.leggilista(p) == codOggetto) {
29. scadenze[j] = 30;
30. libriInPrestito.canclista(p);
31. oggetti.set\_luogo(og, luogo\_attuale);
32. cout << "Fatto.\nhai restituito " << oggetti.get\_oggetto(og).get\_nome() << ".\n"; //   scadenze[p]=30;
33. p = libriInPrestito.succlista(p);
34. j++;
35. }
36. }
37. }
38. }
39. } else lasciato = false;
40. return lasciato;
41. }

#### scadenza()

Il metodo scadenza() permette di effettuare automaticamente la restituzione dei libri nel caso in cui l’utente si dimenticasse di restituire l’oggetto libro.

1. //astro.cpp
2. void Astro::scadenza() {
3. Lista < int > ::posizione p = libriInPrestito.primolista();
4. for ( int j = 0;
5. ((!libriInPrestito.finelista(p)) && (j < MAX\_LIBRI)); p = libriInPrestito.succlista(p)) {
6. int oggetto = libriInPrestito.leggilista(p);
7. int libro = oggetto - 90;
8. int scadenza = --scadenze[libro];
9. if (scadenza == 10) {
10. for (i = 1; i <= oggetti.get\_n\_oggetti(); i++) {
11. if (oggetti.get\_oggetto(i).get\_codice() == oggetto) {
12. cout << "AVVISO: Mancano 10min per la restituzione del libro " << "\n        Corri in Biblioteca!\n";
13. break;
14. }
15. }
16. } else if (scadenza == 0) {
17. scadenze[libro] = 100;
18. for (i = 1; i <= oggetti.get\_n\_oggetti(); i++) {
19. if (oggetti.get\_oggetto(i).get\_codice() == oggetto) {
20. oggetti.set\_luogo(i, 38);
21. cout << "\nIl" << oggetti.get\_oggetto(i).get\_nome() << " e' tornato in Biblioteca. \n\n";
22. cout << "\nA causa della mancata restituzione hai perso 10 minuti del tuo tempo.";
23. break;
24. }
25. }
26. libriInPrestito.canclista(p);
27. tempo = tempo - 10; //PENALITA'
28. }
29. j++;
30. }
31. }

#### aggiorna\_specifiche()

Il metodo aggiorna\_specifiche() permette l’aggiornamento ovvero il decremento della scadenza ad ogni azione.

1. //gioco.h // VIRTUAL
2. virtual void setCabina(int); //\*
3. virtual void setParete(int); //\*
4. virtual int getCabina(); //\*
5. virtual int getParete(); //\*
6. virtual void aggiorna\_specifiche(); //biblioteca
7. virtual bool esegui\_specifiche(int, Mappa & );
8. //gioco.cpp
9. void Gioco::aggiorna\_specifiche() {  }
10. //astro.h
11. #define MAX\_LIBRI 100
12. #define TEMPO 50
13. #include "Portafoglio.h" //MODIFICA D-R(Pisani):Portafoglio + Carta di Credito
14. #include "CartadiCredito.h" //MODIFICA D-R(Pisani):Portafoglio + Carta di Credito
15. class Astro: public Gioco {
17. public:  Astro();
18. virtual~Astro();
19. int  getParete();
20. int  getCabina();
21. void  setParete(int);
22. void  setCabina(int);
23. void  aggiorna\_specifiche();  //biblioteca
25. private:
26. //inizio modifiche biblioteca scatigna
27. int scadenze[10] = {TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO,TEMPO };
28. int i;
29. 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. }

#### ciak(Mappa & M, ifstream & )

Aggiunta del metodo aggiorna\_specifiche() in ciak(Mappa & M, ifstream & ) nel file gioco.cpp

|  |
| --- |
| 1. **void** Gioco::ciak(Mappa &M, ifstream&) 2. { 4. string comando; 5. Luogo l; 6. Stato s; //\* 7. mappa = M; 8. **bool** si;  11. //Pila <string>\* p = new Pila <string>(); 12. Pila <stato\_comando>\* p = **new** Pila <stato\_comando>();  15. **do** 16. { 17. svuotaPila(\*p); 18. init(); 19. introduzione(); 21. initPers(&insPersonaggi); 23. interfaccia.pausa(); // <SP> 24. si = attiva\_enigmi(); // MODIFICA D-R(D-R) 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) + ".";  //Modifica PMF(storia) 37. storia\_gioco.insStoria(stringa\_comando, stringa\_risposta); 38. } 39. l=mappa.get\_luogo(luogo\_attuale); 40. **if**((!Luogo\_p.pilavuota()) && (Codice\_luogo.leggipila()!=luogo\_attuale))                                                                                     //MODIFICA POLIMENA 41. { 42. cout<<"Provieni dal luogo: "<< Codice\_luogo.leggipila() << ": "<< Luogo\_p.leggipila();                    // MODIFICA POLIMENA 43. } 44. Luogo\_p.inpila(mappa.get\_nome\_luogo(luogo\_attuale));                                                            //MODIFICA POLIMENA 45. Codice\_luogo.inpila(luogo\_attuale);                                                                             //MODIFICA POLIMENA 47. **if**(luogo\_attuale == 7) 48. { 49. **if**(oggetti.get\_oggetto(4).get\_luogo() != 0 || oggetti.get\_oggetto(11).get\_luogo()!= 0) 50. bacheca.NuovoMessaggioGioco("\*non uscire fuori dall'astronave se non 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(l); 64. personaggio\_entra(); //  modifica zagaria 66. //if () 67. interfaccia.elenca\_oggetti(oggetti.posizionati\_in(luogo\_attuale), "Vedo:"); 68. //else 69. aggiorna\_premi();//modifiche Francesco Cosma - Premi 70. aggiorna\_specifiche();//biblioteca 71. aggiorna\_tempo(); 72. comando = interfaccia.leggi\_comando(); 73. interpreta(comando,M,p,s,si); 74. aggiorna\_progresso();//modifiche Francesco Cosma - Premi 75. } 76. **while**(!fine\_partita); 77. } 78. **while** (riparti); 79. } |

#### 

### 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.

1. 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("Presa! 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 INTERMEDIE
45. interfaccia.scrivi("Fatto.");
46. } // modifica Gallone - spostamento parentesi per correzione di un errore che si generava nel momento in cui // si prendeva un oggetto che non poteva esser preso
47. bacheca.CancellaMessaggioGioco("\*non uscire fuori dall'astronave se non hai l'equipaggiamento da astronauta");
48. }

Problema: errore di compilazione in caricastatodialogo(string nome)

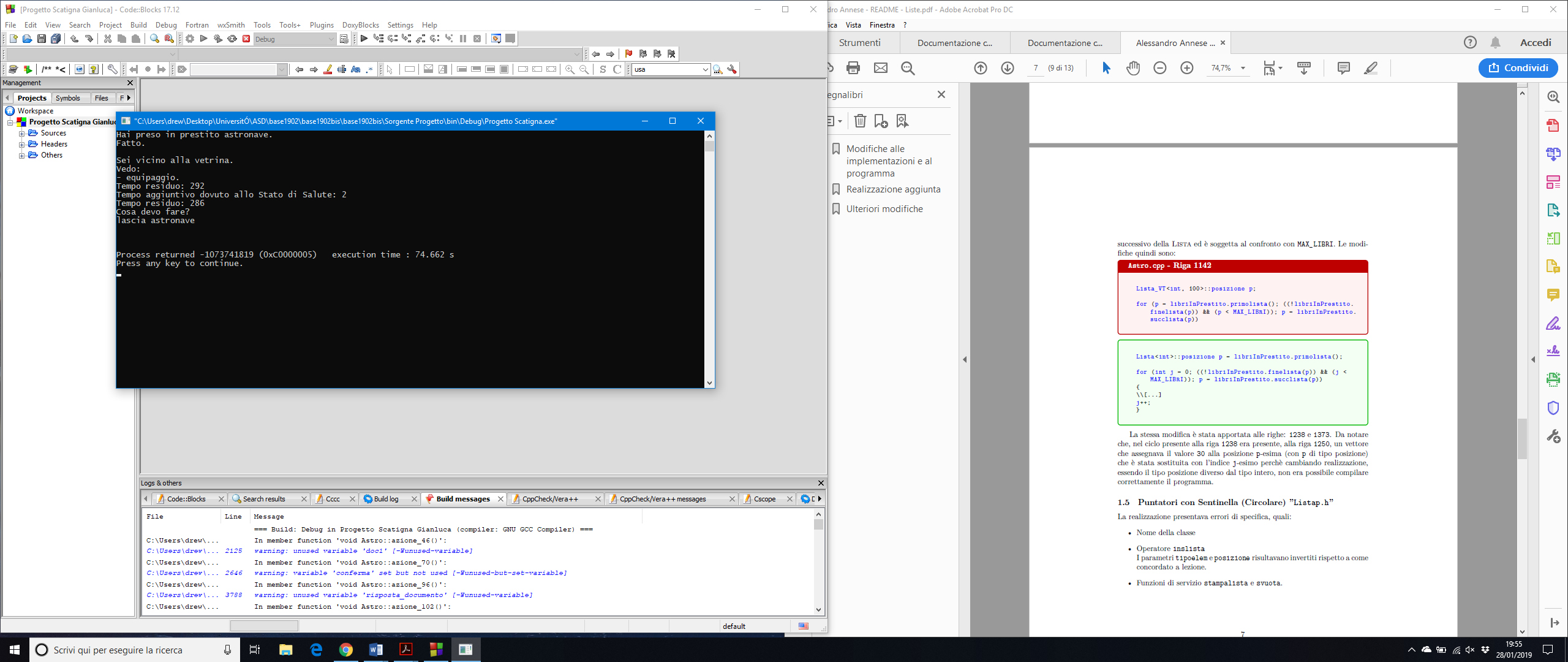
**Causa**: nel metodo **caricastatodialogo(string nome)** il controllo di esistenza del file Statidialoghi.txt in gioco.cpp causa un errore di compilazione (riga 5).

**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. }

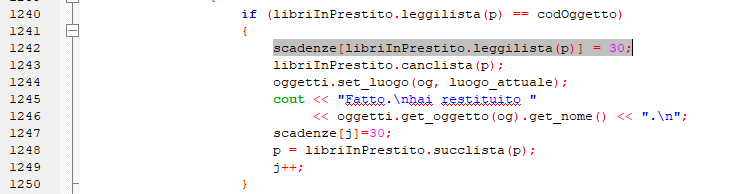
**Correzione Funzionalità lascia libro (Andrea Tursi)**

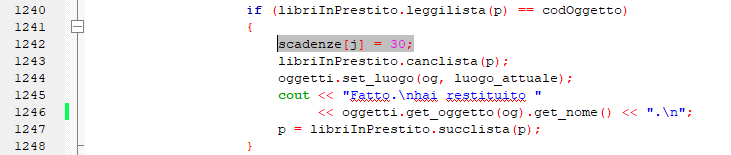
È stato riscontrato un errore nel lasciare i due libri presi in prestito dalla vetrina della biblioteca. L’errore causava l’interruzione improvvisa del gioco.

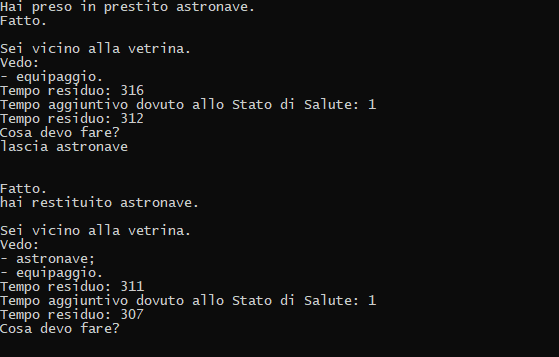


Il problema è stato risolto andando a modificare il file Astro.cpp, dove nel metodo **lascia\_specifiche()**, alla riga 1242, veniva assegnato il valore intero 30 alla posizione p-esima (con p di tipo posizione lista) del vettore scadenze.

Assegnazione che veniva ripetuta correttamente alla riga 1247, utilizzando l’indice j creato appositamente.







## 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

### 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** (NNSSEEOOUUDD)
* **Tipo di strada** (via 6,8,8)

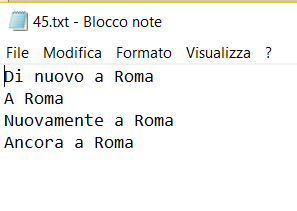
È stato modificato il codice direzioni al luogo 13 “**Stazione di servizio**” per arrivare a **Bari** digitando *Est*:

* Narracci: 000046000004

### Descrizioni(cartella)

Nella cartella Descrizioni sono stati importati tre file .txt dal progetto di Narracci.

Ne riporto solo un esempio.



### 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;

### Astro.h

* Prototipo azione\_80 “*guarda automobile*”

void azione\_80(); //Modifica DELLA FOLGORE GRAZIA - Guarda automobile

### 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

}

### 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

### 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

### Luogo.h

* Prototipo funzione di controllo

bool leggi\_predefinito(); //Modifica DELLA FOLGORE GRAZIA

### 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

### Oggetti.h

* Prototipo metodo per cambio posizione oggetti

void set\_luogo\_oggetti\_pos\_in(int, int); //Modifica DELLA FOLGORE GRAZIA

### 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

### 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;}

### 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

### 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

### 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\_ \*/

### 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;

}

### 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 \*/

### 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

### 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\_ \*/

### 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){

}

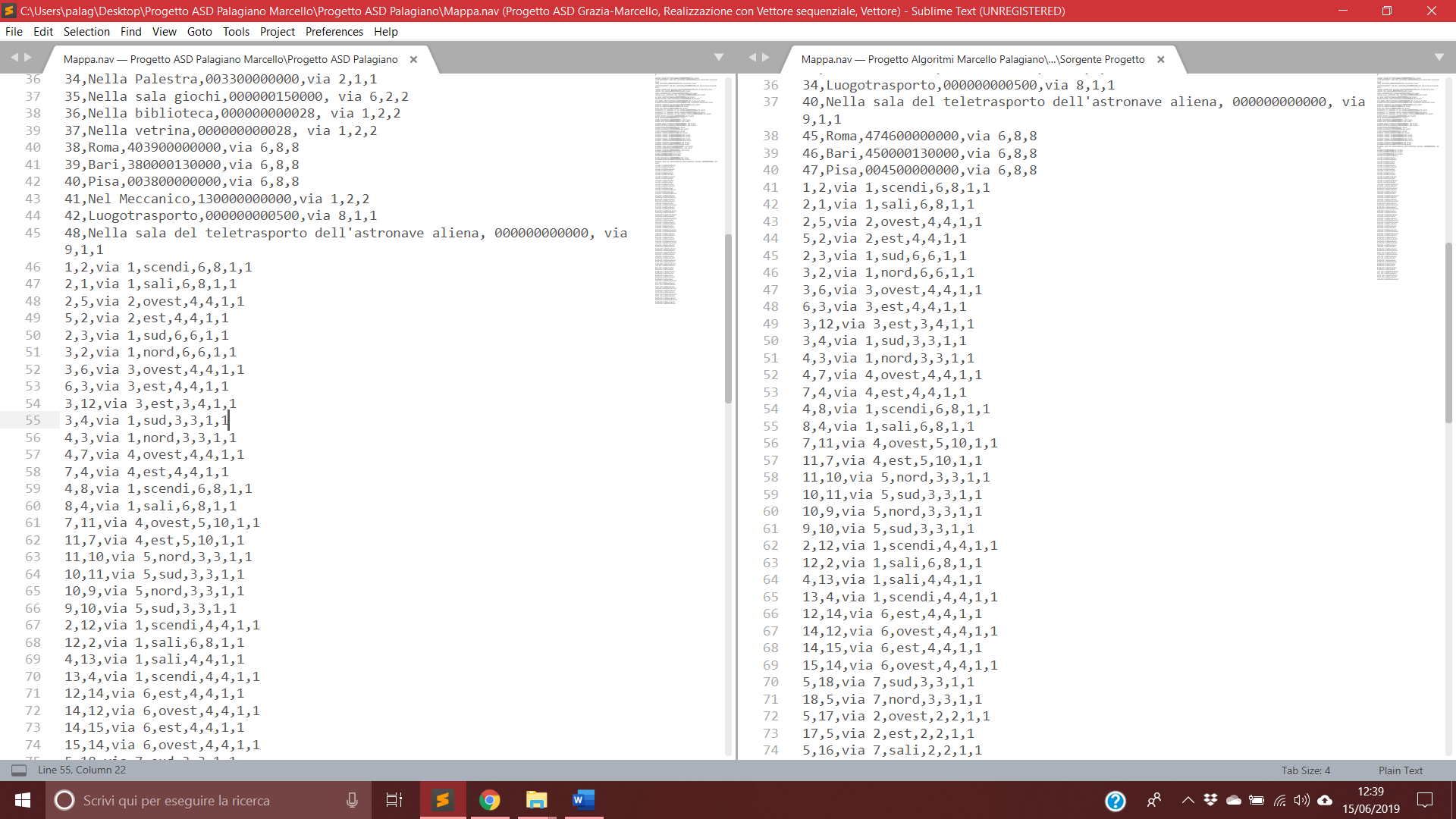
Automobile::~Automobile(){

}

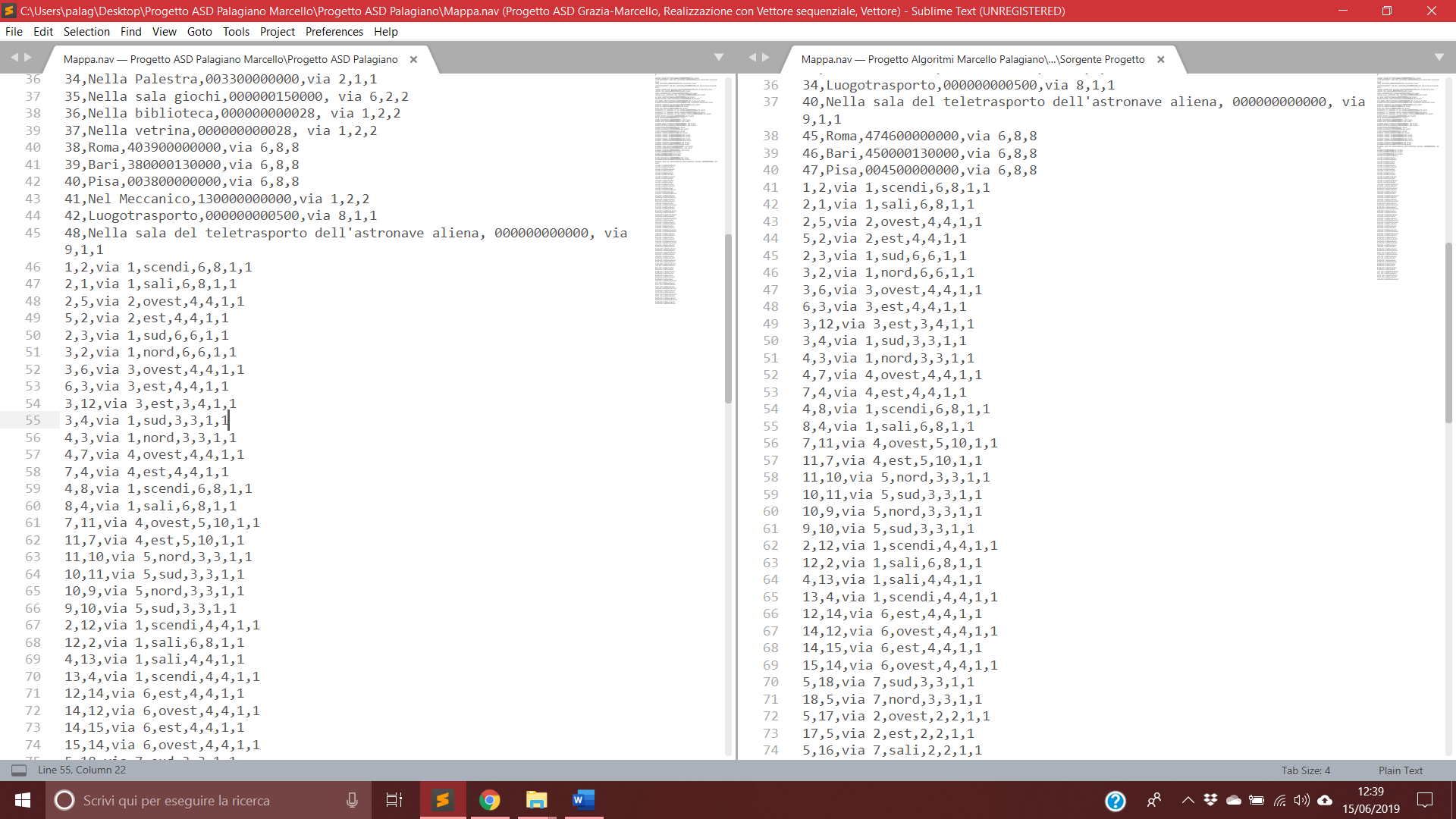
## Modifica codici luoghi – Palagiano Marcello

### Modifica file “Mappa.nav”

Il codice nel progetto base1906 era il seguente:



Il nuovo codice è:



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,000046000004,via 1,8,8

15,In una banca,230000140000,via 6,2,2

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

33,Nella Stazione,001200000000,via 2,1,1

5,34,via 8,scendi,1,1,1,1

Il nuovo codice è:

13,Nella Stazione di Servizio,000039000004,via 1,8,8

15,In una banca,230035140000,via 6,2,2

28,Nella Scuola,300129313600,via 1,1,1

33,Nella Stazione,341200000000,via 2,1,1

5,42,via 8,scendi,1,1,1,1

33,34,via 2,nord,2,2,1,1

34,33,via 2,sud,2,2,1,1

35,15 via 6,ovest,2,2,1,1

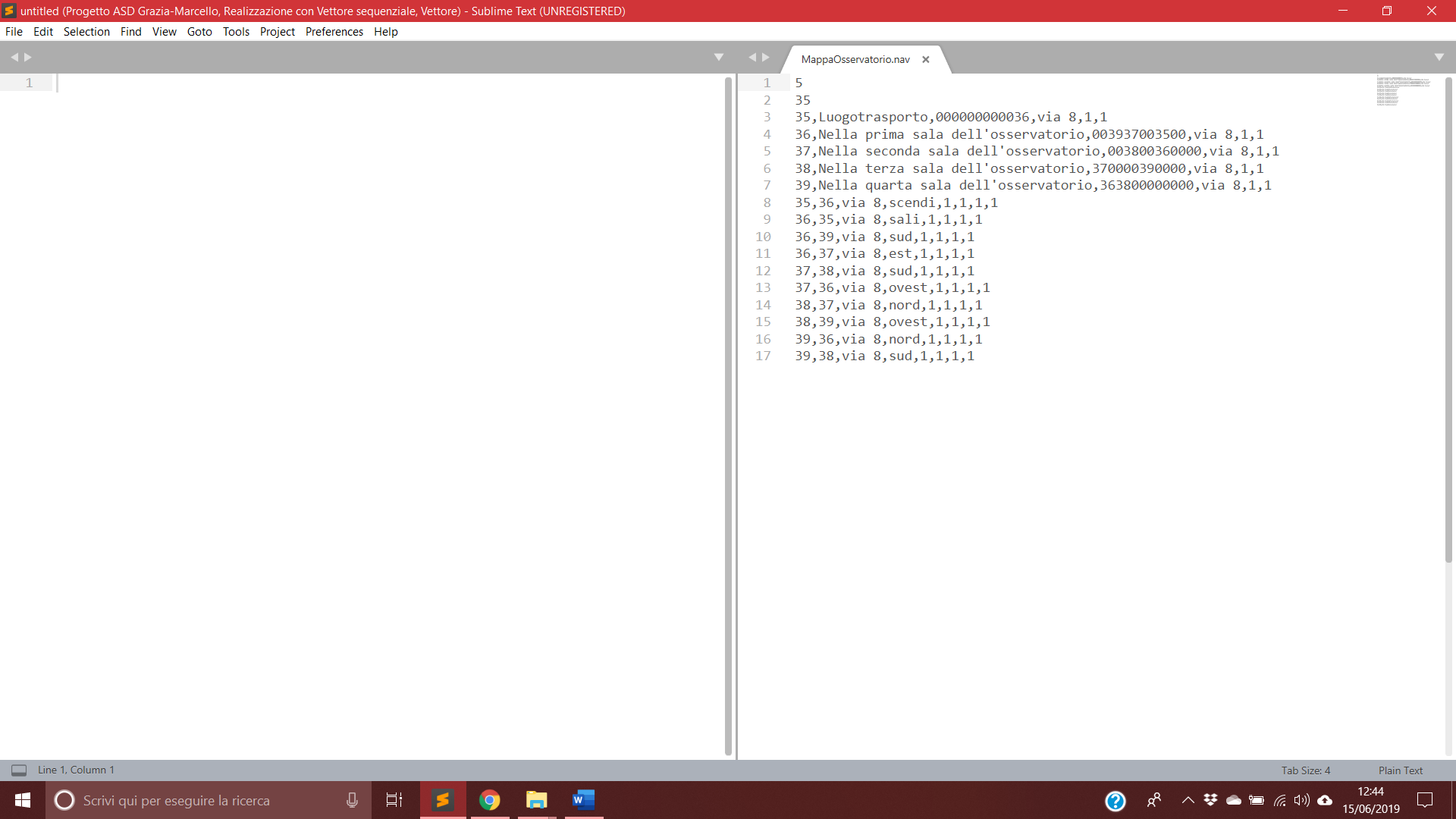
15,35 via 6,est,2,2,1,1

28,36,via 2,sali,2,2,1,1

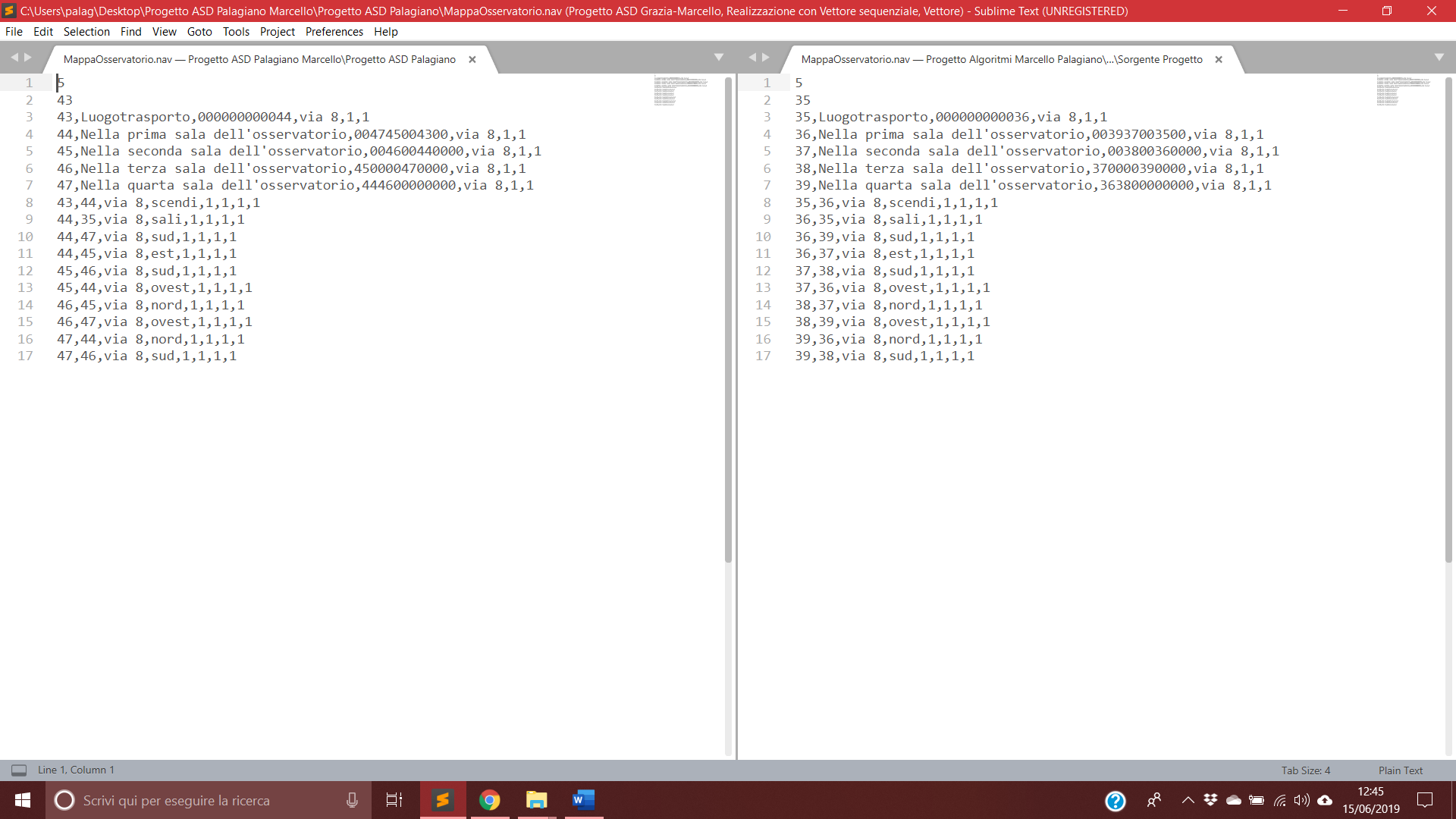
36,28,via 2,scendi,2,2,1,1

### Modifica file “MappaOsservatorio.nav”

Il codice nel progetto base1906 era il seguente:

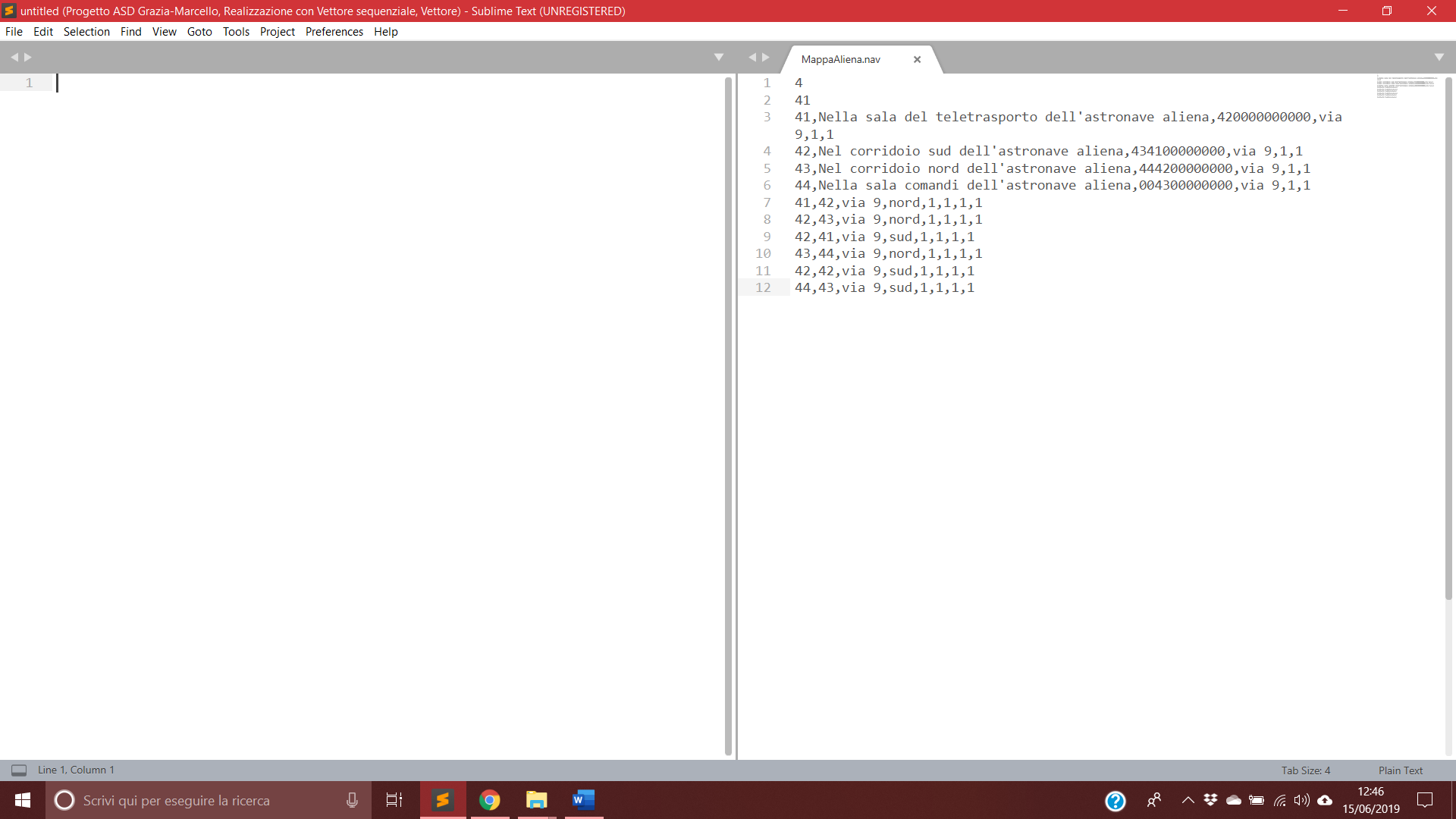


Il nuovo codice è:

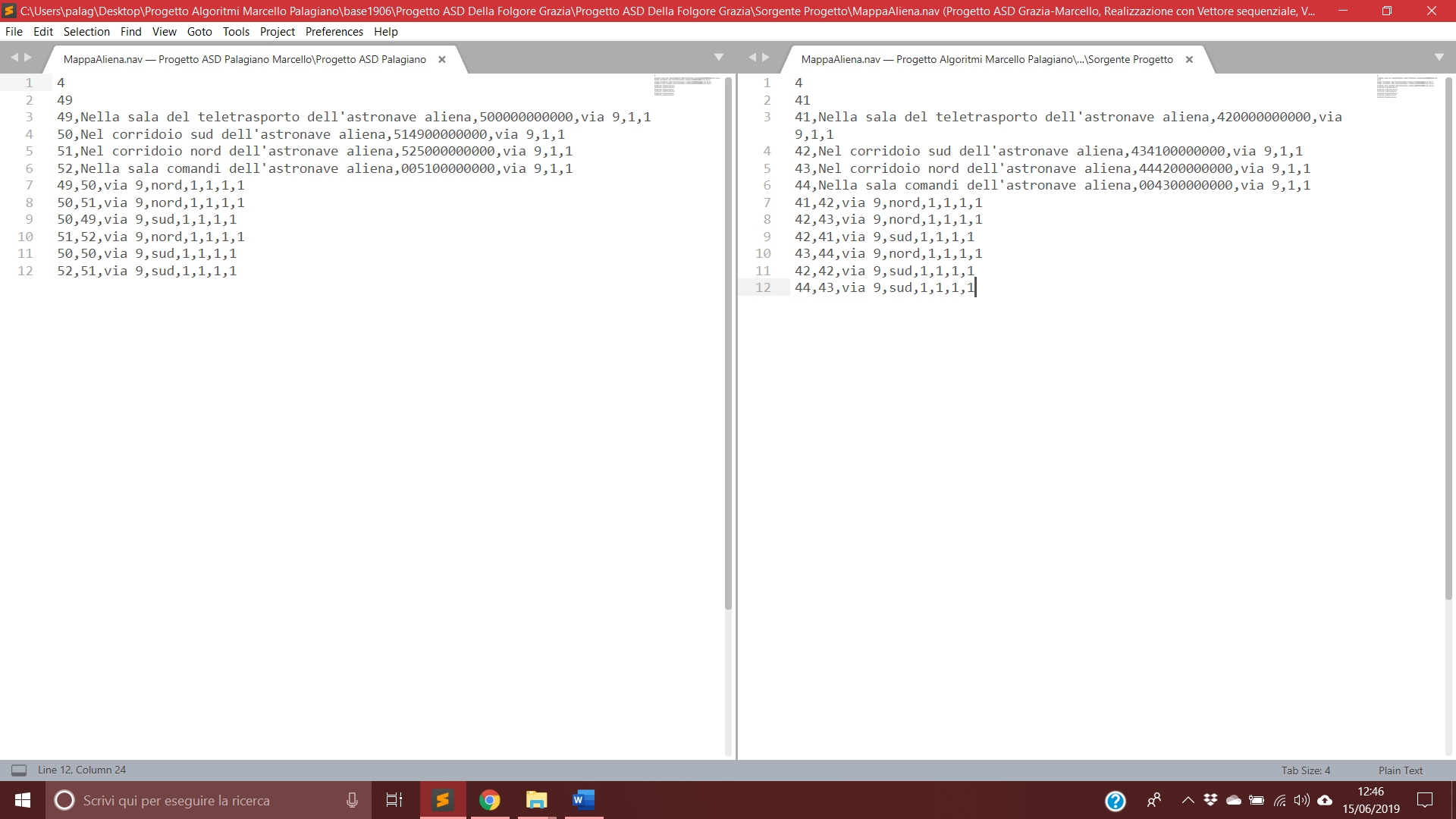


### Modifica file “MappaAliena.nav”

Il codice nel progetto base1906 era il seguente:

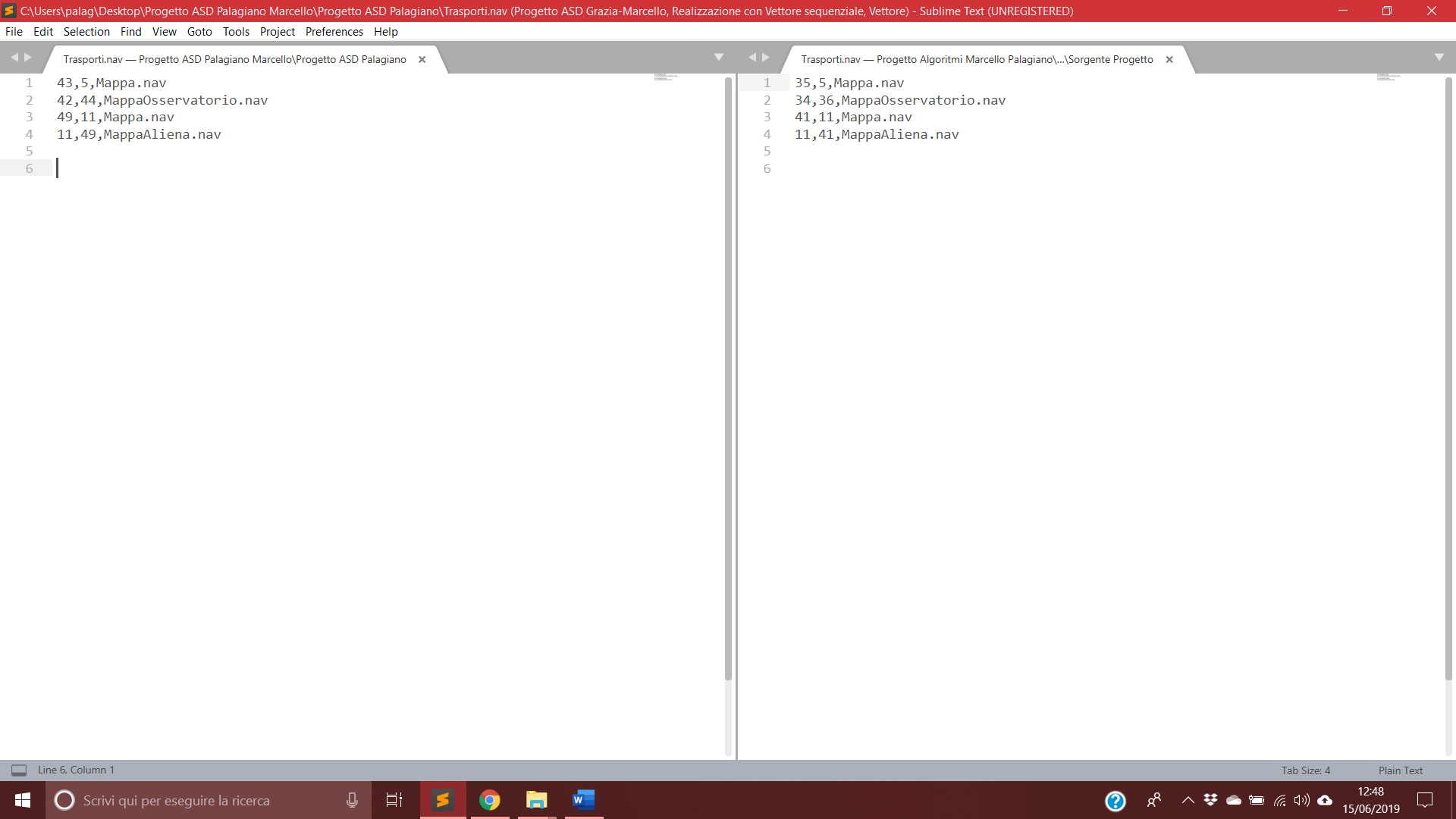


Il nuovo codice è:

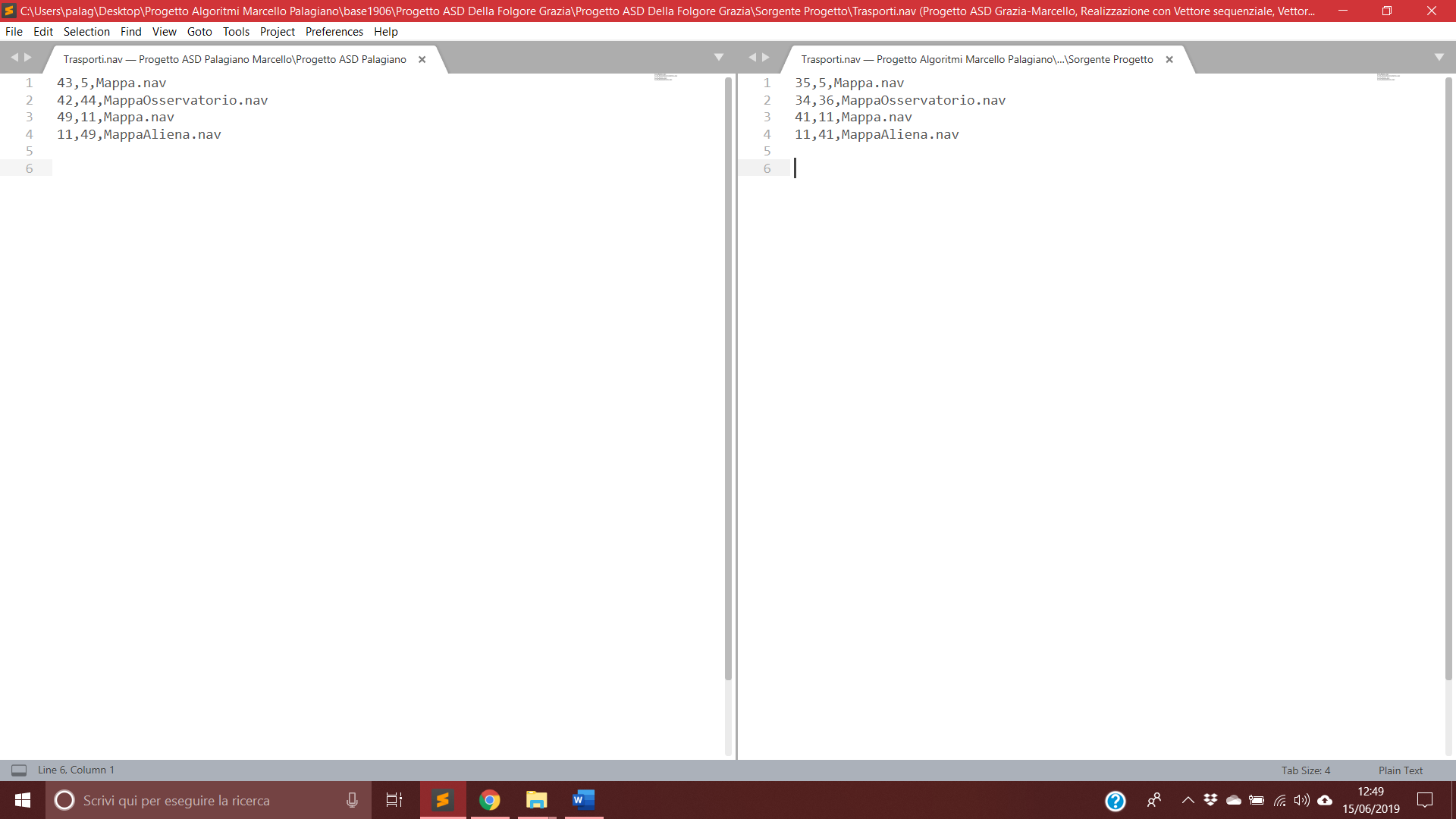


### Modifica file “Trasporti.nav”

Il codice nel progetto base1906 era il seguente:



Il nuovo codice è:



### Modifica righe di codice

Sono state modificate le righe di codice relative ai suddetti luoghi per i quali si è dovuto modificare il codice luogo.

azioni.inserisci(4900290048, 118); //Se si usa il teletrasporto nell'astronave aliena -- Palagiano Marcello: modifica codice luogo da 40 a 48

//INIZIO modifiche DAVIDE MANTELLINI -- Palagiano Marcello: modifica codice luogo da 35 a 34

azioni.inserisci(3400290057, 120); //usa panca

azioni.inserisci(3400290047, 121); //usa tapis

azioni.inserisci(3400290099, 122); //usa terminale

azioni.inserisci(3400100099, 122); //guarda terminale

azioni.inserisci(3400290084, 123); //usa schede

azioni.inserisci(3400290097, 124); //usa macchinetta

//FINE modifiche DAVIDE MANTELLINI

//INIZIO modifiche GIACOMO CICALA -- Palagiano Marcello: modifica codice luogo da 36 a 35

azioni.inserisci(3500290086, 125); //usa slotmachine

//FINE modifiche GIACOMO CICALA

//inizio modifiche biblioteca Scatigna -- Palagiano Marcello: modifica codice luogo da 37 a 36 (biblioteca), da 38 a 37 (vetrina)

azioni.inserisci(3600250070, 126); //nuova azione etichetta su biblioteca

azioni.inserisci(3600220019, 127); //nuova azione di apertura su vetrina

azioni.inserisci(3700930019, 128); //nuova azione di chiusura su vetrina

azioni.inserisci(250090, 129); //leggi libro astronave

azioni.inserisci(250091, 129); //leggi libro equipaggio

//fine modifica biblioteca Scatigna

oggetti.inserisci(Oggetto("un teletrasporto",48,-49)); //-- Palagiano Marcello: modifica codice luogo da 41 a 49

//INIZIO modifiche DAVIDE MANTELLINI -- Palagiano Marcello: modifica codice luogo da 35 a 34

oggetti.inserisci(Oggetto("una panca per gli addominali", 57, -34));

oggetti.inserisci(Oggetto("un tapis roulant", 47, -34));

oggetti.inserisci(Oggetto("un terminale informativo",99, -34));

oggetti.inserisci(Oggetto("delle schede di allenamento", 84, 34));

oggetti.inserisci(Oggetto("una macchinetta",97, -34));

//FINE modifiche DAVIDE MANTELLINI

//INIZIO modifiche GIACOMO CICALA

oggetti.inserisci(Oggetto("una slotmachine", 86,-35));

oggetti.inserisci(Oggetto("5 euro",90,-99)); //Luogo -99 in quanto non visibili

oggetti.inserisci(Oggetto("10 euro",91,-99));

oggetti.inserisci(Oggetto("20 euro",92,-99));

oggetti.inserisci(Oggetto("50 euro",93,-99));

oggetti.inserisci(Oggetto("100 euro",94,-99));

//Fine modifiche GIACOMO CICALA

//inizio modifiche biblioteca Scatigna -- Palagiano Marcello: modifica codice luogo da 37 a 36 (biblioteca), da 38 a 37 (vetrina)

oggetti.inserisci(Oggetto("etichetta", 70, -36));

oggetti.inserisci(Oggetto("vetrina",19, -36));

oggetti.inserisci(Oggetto("astronave", 90, 37));

oggetti.inserisci(Oggetto("equipaggio", 91, 37));

//fine modifica Scatigna

#### 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.finelista(p)) && (j < MAX\_LIBRI)); p =libriInPrestito.succlista(p))

{

int oggetto = libriInPrestito.leggilista(p);

//cout << "oggetto: " << oggetto << " \n";

int libro = oggetto - 90;

//cout << "libro : " << libro << " \n";

int scadenza = --scadenze[libro];

//cout << scadenza << " \n";;

if (scadenza == 10)

{

for (i = 1; i <= oggetti.get\_n\_oggetti(); i++)

{

if (oggetti.get\_oggetto(i).get\_codice() == oggetto)

{

cout << "AVVISO: Mancano 10min per la restituzione del libro "

<< "\n Corri in Biblioteca!\n";

break;

}

}

}

else if (scadenza == 0)

{

scadenze[libro] = 100;

for (i = 1; i <= oggetti.get\_n\_oggetti(); i++)

{

if (oggetti.get\_oggetto(i).get\_codice() == oggetto)

{

oggetti.set\_luogo(i, 37);

cout << "\nIl " << oggetti.get\_oggetto(i).get\_nome() << " e' tornato in Biblioteca. \n\n";

cout << "\nA causa della mancata restituzione hai perso 10 minuti del tuo tempo.";

break;

}

}

libriInPrestito.canclista(p);

tempo = tempo - 10; //PENALITA

}

j++;

}

}

//Fine modifiche Biblioteca Scatigna

//inizio modifiche biblioteca Scatigna -- Palagiano Marcello: modifica codice luogo da 37 a 36 (biblioteca), da 38 a 37 (vetrina)

void Astro::azione\_126()

{

interfaccia.scrivi("\n ---- REGOLAMENTO ------");

interfaccia.scrivi("\n\n1) E' possibile prendere un libro in "

"\nprestito e restituirlo entro e non oltre 3 ore");

interfaccia.scrivi("\n2) La restituzione e' valida solo se i libri "

"\nvengono lasciati al posto originario.");

interfaccia.scrivi("\n3) Si consiglia di non lasciare i libri sparsi"

"\nin biblioteca");

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

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

#### 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);

}

## 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

### 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

### 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;

}

### 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

### 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;

}

### 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

### 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

### 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

### 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*.

#### 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

#### 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

#### 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

#### 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

#### 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

#### Implementazione procedure per le azioni 130-139

##### **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");

}

##### **Azione 131 Guarda ricambio**

void Astro::azione\_131() //esamina ricambio astronave

{

interfaccia.scrivi("Vedo un pezzo di ricambio per l'astronave... Purtroppo e' rotto!");

}

##### **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();

}

}

}

##### **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();

}

}

##### **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");

}

##### **Azione 135 Guarda diario**

void Astro::azione\_135() //guarda diario

{

interfaccia.scrivi("Sembra un diario delle attivita'. Magari potrebbe essere utile leggerlo!");

}

##### **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;

}

}

}

##### **Azione 137 Guarda cartello**

void Astro::azione\_137() //guarda cartello

{

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

cout<<"\* OFFICINA. Per qualsiasi richiesta rivolgersi al personale autorizzato! \*\n";

cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n";

}

##### **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);

}

}

##### **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!");

}

}

else

interfaccia.scrivi("Hai gia' cambiato la batteria dell'automobile!");

}

//FINE integrazione PALAGIANO MARCELLO

#### 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

### 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

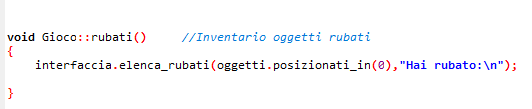
41,13,via 1,nord,1,1,1,1

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

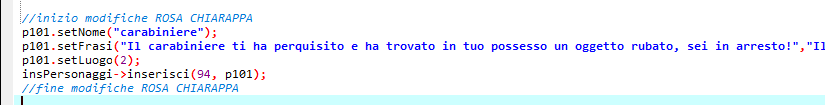
### Verbo “ruba”

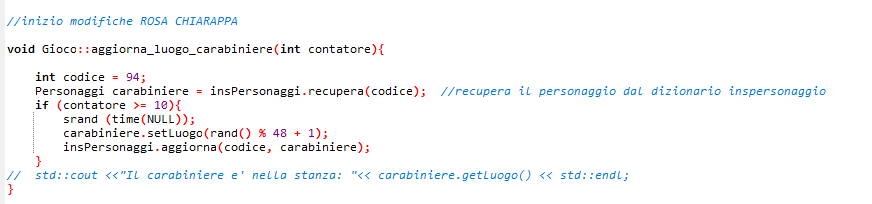


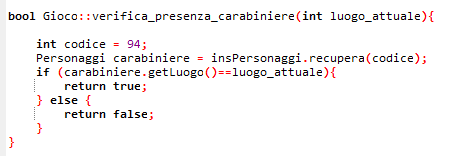
### Verbo “rubati”

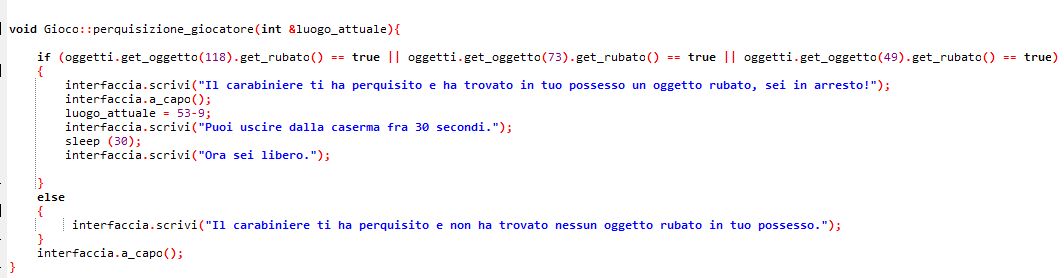


### Personaggio”carabiniere”









### Luogo “caserma”



## Zaino e valigia

### Astro.h

Sono state aggiunte le seguenti funzioni:



### Astro.cpp

Sono stati aggiunti i seguenti vocaboli:

vocabolario.inserisci("valigia", 155);

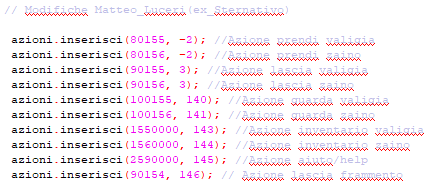
vocabolario.inserisci("zaino", 156);

vocabolario.inserisci("raccogli", 258);

vocabolario.inserisci("aiuto", 259);

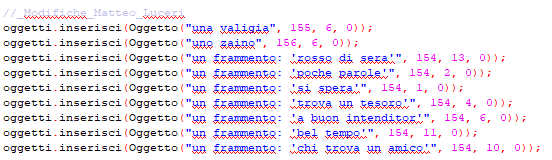
vocabolario.inserisci("help", 259);

Sono stati aggiunte le seguenti azioni:

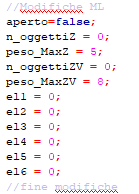


Sono state modificate le seguenti azioni:



In tutti gli oggetti è stato aggiunto un attributo “peso” e sono stati aggiunti:

Sono state inizializzate le seguenti variabili:



Sono state modificate le seguenti funzioni:

Le funzioni preso\_specifiche e prendi\_specifiche sono state “spostate” (con le dovute modifiche elencate di seguito) in indosso\_specifiche e indossa\_specifiche e le prime due sono state modificate in questo modo:

Immagine che contiene screenshot

Descrizione generata automaticamente

Sono state aggiunte le seguenti azioni:

**void** Astro::azione\_140(){ //Azione guarda valigia

interfaccia.scrivi("E' la valigia del secondo pilota.");

interfaccia.scrivi("Sembra molto capiente.");

}

**void** Astro::azione\_141(){ //Azione guarda zaino

interfaccia.scrivi("E' il tuo zaino.");

interfaccia.scrivi("Puo' esserti utile per trasportare oggetti.");

}

Immagine che contiene screenshot

Descrizione generata automaticamente

**void** Astro::azione\_142(){ //Azione indossa/metti

**if** (oggetti.get\_oggetto(og).get\_luogo() == 0)

// controllo che l'oggetto si trovi nell'inventario

interfaccia.scrivi("- Gia' fatto.");

**else if** (oggetti.get\_oggetto(og).get\_luogo() < 0)

// controllo che l'oggetto non sia indossabile

interfaccia.scrivi("- Non e' possibile.");

**else if** (!indossa\_specifiche()) {

**if**(oggetti.get\_oggetto(144).get\_luogo() == 0){

**if** (oggetti.get\_oggetto(4).get\_luogo() == 20 ||

oggetti.get\_oggetto(11).get\_luogo() == 20 ||

oggetti.get\_oggetto(22).get\_luogo() == 20){

**if**(og == 4 || og == 11 || og == 23){

**int** c;

c=z.leggipila();

**if**(og == oggetti.get\_zaino2(c)){

// controllo che l'oggetto sia proprio il primo oggetto nello zaino

oggetti.set\_luogo(og,0);

z.fuoripila();

n\_oggettiZ--;

peso\_MaxZ += oggetti.get\_oggetto(og).get\_peso();

}

**else if**(oggetti.get\_oggetto(og).get\_luogo() == luogo\_attuale){

// controllo che l'oggetto si trovi nel luogo attuale

oggetti.set\_luogo(og,0);

}

**else**{

interfaccia.scrivi("Devi prima lasciare: ");

oggetti.get\_zaino(c);

}

}**else**{

oggetti.set\_luogo(og,0);

interfaccia.scrivi("Fatto.");

}

}**else**{

oggetti.set\_luogo(og,0);

}

} **else if**(oggetti.get\_oggetto(143).get\_luogo() == 0){

**if** (oggetti.get\_oggetto(4).get\_luogo() == 20 ||

oggetti.get\_oggetto(11).get\_luogo() == 20 ||

oggetti.get\_oggetto(23).get\_luogo() == 20){

**if**(og == 4 || og == 11 || og == 23){

**int** c;

c=og;

**if**(ins.Appartiene(oggetti.get\_valigia(c))){

// controllo che l'oggetto sia presente nella valigia

oggetti.set\_luogo(og,0);

ins.Cancella(oggetti.get\_valigia(c));

n\_oggettiZV--;

peso\_MaxZV += oggetti.get\_oggetto(og).get\_peso();

}

**else if**(!ins.Appartiene(oggetti.get\_valigia(c)))

//controllo che l’oggetto non sia presente nella valigia

interfaccia.scrivi("- Non ce l'hai in valigia.");

**else if**(oggetti.get\_oggetto(og).get\_luogo() == luogo\_attuale){

//controllo che l’oggetto si trovi nel luogo attuale

oggetti.set\_luogo(og,0);

}

}

**else**{

oggetti.set\_luogo(og,0);

interfaccia.scrivi("Fatto.");

}

}

**else**{

oggetti.set\_luogo(og,0);

}

}

**else**{

**if**(og != 0 && oggetti.get\_oggetto(og).get\_luogo() != 20){

// controllo che l’oggetto non si trovi in un altro luogo e che l’oggetto non si trovi nello zaino o nella valigia

oggetti.set\_luogo(og,0);

}

**else**

interfaccia.scrivi("- Sta nello zaino o nella valigia, e non li hai.");

}

**if** (!indosso\_specifiche()){ // faccio alcuni controlli in indosso\_specifiche

**if**(oggetti.get\_oggetto(og).get\_luogo() == 0)

//controllo che l’oggetto si trovi nell’inventario

interfaccia.scrivi("Fatto.");

}

}

}

**void** Astro::azione\_143(){ //Azione inventario valigia

**if**(oggetti.get\_oggetto(143).get\_luogo() == 0){

**if** (!ins.InsiemeVuoto()){

interfaccia.scrivi("Inventario Valigia");

interfaccia.scrivi("\nVedo: ");

**if**(ins.Appartiene(50)) // controllo che il casco sia presente nella valigia

oggetti.get\_zaino(50);

**if**(ins.Appartiene(51)) // controllo che la tua sia presente nella valigia

oggetti.get\_zaino(51);

**if**(ins.Appartiene(52)) // controllo che il camice sia presente nella valigia

oggetti.get\_zaino(52);

**if**(ins.Appartiene(55)) // controllo che il manuale sia presente nella valigia

oggetti.get\_zaino(55);

**cout** << "\nTotale Oggetti nella Valigia: " << n\_oggettiZV << **endl**;

**cout** << "Spazio disponibile: "<< peso\_MaxZV << " su 8 kg."<< **endl**;

}

**else**

interfaccia.scrivi("E' vuota.");

}

**else**

interfaccia.scrivi("Non ce l'hai.");

}

**void** Astro::azione\_144(){ //Azione inventario zaino

**if**(oggetti.get\_oggetto(144).get\_luogo() == 0){

// controllo che l'oggetto zaino si trovi nell'inventario

**if** (!z.pilavuota()) { // controllo che l'oggetto zaino non sia vuoto //\*

interfaccia.scrivi("Inventario Zaino");

interfaccia.scrivi("\nVedo in cima: ");

**int** c;

c=z.leggipila(); //\*

oggetti.get\_zaino(c);

**cout** << "\nTotale Oggetti nello Zaino: " << n\_oggettiZ << **endl**;

**cout** << "Spazio disponibile: "<< peso\_MaxZ << " su 5 kg."<< **endl**;

}

**else**

interfaccia.scrivi("E' vuoto.");

}

**else**

interfaccia.scrivi("Non lo hai.");

}

**void** Astro::azione\_145(){ //Azione aiuto/help

interfaccia.scrivi("\nCOMANDI DI GIOCO:");

interfaccia.scrivi("\nDirezioni: ");

interfaccia.scrivi("- n/nord: per muoverti in avanti;");

interfaccia.scrivi("- s/sud: per muoverti indietro;");

interfaccia.scrivi("- e/est: per muoverti a destra;");

interfaccia.scrivi("- w/o/ovest: per muoverti a sinistra;");

interfaccia.scrivi("- a/alto/sali: per salire ad un piano superiore;");

interfaccia.scrivi("- b/basso/scendi: per scendere ad un piano inferiore;");

interfaccia.scrivi("\nAzioni: ");

interfaccia.scrivi("- prendi/raccogli: per trasportare un oggetto in mano   
 o con se(zaino/valigia);");

interfaccia.scrivi("- indossa/metti: per indossare un oggetto(es. casco);");

interfaccia.scrivi("- guarda: per guardare ed ottenere informazioni su un oggetto (es.tuta);");

interfaccia.scrivi("- lascia/togli/leva: per lasciare o togliersi gli oggetti trasportati;");

interfaccia.scrivi("- apri: per aprire un oggetto fisso(es. armadietto);");

interfaccia.scrivi("- leggi: per leggere una scritta(es.cartello);");

interfaccia.scrivi("- spingi/tira: per spingere o tirare un oggetto fisso(es.leva);");

interfaccia.scrivi("- premi/schiaccia: per premere un oggetto fisso(es.pulsante);");

interfaccia.scrivi("- inventario/cosa: per accedere all'inventario degli oggetti trasportati;");

interfaccia.scrivi("- zaino: per accedere agli oggetti trasportati nello zaino;");

interfaccia.scrivi("- valigia: per accedere agli oggetti trasportati nella valigia;");

interfaccia.scrivi("- save/load: per salvare o caricare la partita;");

interfaccia.scrivi("- mappa/navigatore: per avviare il navigatore SIMUNAV;\n"); }

**bool** Astro::esegui\_specifiche(**int** a, Mappa &M) -> sono stati aggiunti i seguenti case:

**case** 140:

azione\_140();

**break**;

**case** 141:

azione\_141();

**break**;

**case** 142:

azione\_142();

**break**;

**case** 143:

azione\_143();

**break**;

**case** 144:

azione\_144();

**break**;

**case** 145:

azione\_145();

**break**;

### Gioco.h

È stata aggiunta l’include per permettere l’implementazione della valigia:

#include "Valigia.h"

Sono state aggiunte le seguenti variabili:

Pila<**int**> p;

Pila<**int**> z;

**int** n\_oggettiZ; //Numero di oggetti nello zaino

**int** peso\_MaxZ; //Peso massimo trasportabile nello zaino

Insieme<**int**> ins;

**int** n\_oggettiZV; //Numero di oggetti nella valigia

**int** peso\_MaxZV; //Peso massimo trasportabile nella valigia

//el1=primo elemento zaino; el2=secondo elemento zaino;

//el3 [..] el6 elementi valigia

**int** el1;

**int** el2;

**int** el3;

**int** el4;

**int** el5;

**int** el6;

**bool** salva; //Boooleano usato nella save e nella load

**string** risposta1; //risposta al frammento

**bool** controllo1;

**bool** controllo2;

Lista<**int**> L

### Gioco.cpp

Sono state modificate le seguenti funzioni:

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

### Immagine che contiene screenshot Descrizione generata automaticamente

### Immagine che contiene screenshot Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

### Mappa.h

È stata aggiunta l’include per permettere l’implementazione dello zaino:

#include "Zaino.h"

### Interfaccia.cpp

È stata modificata la seguente funzione:

**void** Interfaccia::elenca\_oggetti(Oggetti oggetti, string premessa)

{

//Modifica Mirco Sternativo

**int** n\_oggettiZ = oggetti.get\_n\_oggettiZ();

**if** (n\_oggettiZ > 0)

{

**cout** << premessa;

**for** (int i = 1; i <= n\_oggettiZ; i++)

{

**cout** << "\n- " << oggetti.get\_oggetto(i).get\_nome();

**if** (i == n\_oggettiZ)

{

cout << ".";

}

**else**

{

cout << ";";

}

}

cout << endl;

}

//fine modifiche

}

### Oggetti.h

Sono stati aggiunti i seguenti metodi:

**int** get\_n\_oggettiZ();

**void** get\_zaino(**int**);

**int** get\_zaino2(**int**);

**int** get\_valigia(**int**);

### Oggetti.cpp

Sono stati aggiunti i seguenti metodi:

**int** Oggetti::get\_n\_oggettiZ()

{

**return** fo;

}

**void** Oggetti::get\_zaino(int c)

{

**bool** trovato = **false**;

**int** i = 1;

**while** (i <= fo && !trovato) {

// scandisco tutti gli oggetti presenti nel vocabolario finché o finiscono gli oggetti o trovato viene impostato a true

**if** (oggetti[i].get\_codice() == c){

// controllo che il codice dell’oggetto con indice “i” sia uguale al codice “c”

**cout** << "- " <<oggetti[i].get\_nome() << **endl**;

trovato = **true**;

}

i++;

}

}

**int** Oggetti::get\_zaino2(int c)

{

**bool** trovato = **false**;

**int** i = 1;

**int** i2 = 0;

**while** (i <= fo && !trovato) {

// scandisco tutti gli oggetti presenti nel vocabolario finché o finiscono gli oggetti o trovato viene impostato a true

**if** (oggetti[i].get\_codice() == c){

// controllo che il codice dell’oggetto con indice “i” sia uguale al codice “c”

i2=i;

trovato = **true**;

}

i++;

}

**return**(i2);

}

**int** Oggetti::get\_valigia(int c)

{

**bool** trovato = **false**;

**int** i = 1;

**int** i2 = 0;

**while** (i <= fo && !trovato) {

// scandisco tutti gli oggetti presenti nel vocabolario finché o finiscono gli oggetti o trovato viene impostato a true

**if** (i == c){//controllo che l’indice “i” sia uguale al codice “c”

i2 = oggetti[i].get\_codice();

trovato = **true**;

}

i++;

}

**return**(i2);

}

Ed è stata modificata la seguente funzione:

**int** Oggetti::luogo\_oggetto(**int** c2, **int** lu) {

**bool** trovato = **false**;

**int** i = 1;

**int** og = 0;

**while** (i <= fo && !trovato) {

**if** (c2 == 30){

**if**(oggetti[i].get\_luogo() == lu){

og = i;

}

}**else if** (oggetti[i].get\_codice() == c2)

**if** (abs(oggetti[i].get\_luogo()) == lu || oggetti[i].get\_luogo() == 0 || oggetti[i].get\_luogo() == 20) {

og = i;

trovato = **true**;

}

i++;

}

**return** og;

}

### Oggetto.h

È stato creato un nuovo costruttore che integra l’attributo peso

Oggetto(**string** n, **int** c, **int** l, **int** w);

//Modifica Mirco Sternativo -- Peso oggetto

È stato modificato il costruttore avente l’attributo prezzo, aggiungendo il peso

Oggetto(**string** n, **int** c, **int** l, **int** w, **float** p );

//MODIFICA D-R(D'Orsi):Negozio + Banca

//Modifica Mirco Sternativo -- Peso oggetto

È stato modificato il metodo get\_peso:

**int** get\_peso(); //Modifica Mirco Sternativo

### Oggetto.cpp

È stato aggiunto il seguente metodo

Oggetto::Oggetto(**string** n, **int** c, **int** l, **int** w) //w: weight

{

nome = n;

codice = c;

luogo = l;

peso = w;

}

Sono stati modificati i seguenti metodi

Oggetto::Oggetto(**string** n, **int** c, **int** l, **int** w, **float** p)

{

nome = n;

codice = c;

luogo = l;

peso = w;

prezzo = p;

}

**int** Oggetto::get\_peso()

{

**return** peso;

}

### Zaino.h

È stato semplicemente aggiunto il file senza alcuna modifica particolare.

### Valigia.h

È stato semplicemente aggiunto il file senza alcuna modifica particolare.

# Strutture intercambiabili

Sono state inserite le strutture intercambiabili:

## Lista

### 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 succlista(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)

//sovraccarichi

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.finelista(ind2))

{

inslista(b.leggilista(ind2),ind);

ind = succlista(ind);

ind2 = b.succlista(ind2);

}

}

template <class tipoelem> Lista<tipoelem>::~Lista() //distruttore

{

posizione ind=primolista();

while (!finelista(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>::leggilista(posizione p) const

{

if (!finelista(p))

return (p->leggicella());

}

template <class tipoelem> void Lista<tipoelem>::scrivilista(tipoelem e, posizione p)

{

if (!finelista(p))

p->scrivicella(e);

}

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

{

typename Lista<tipoelem>::posizione temp;

temp = new Cella<tipoelem>;

temp->scrivicella(e);

temp->scrivisucc(p);

temp->scriviprec(p->leggiprec());

if(p == primolista())

lista = temp;

else

p->leggiprec()->scrivisucc(temp);

p->scriviprec(temp);

p = temp;

}

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

{

if (!finelista(p))

{

posizione temp=p;

if (p==lista)

{

lista=p->leggisucc();

p=lista;

}

else

{

p=p->leggisucc();

(preclista(temp))->scrivisucc(p);

}

delete(temp);

lungh--;

}

}

//MODIFICA D-R(D-R) //

template <class tipoelem> Cella<tipoelem>\* Lista<tipoelem>::get\_posizione()const

{

return lista;

}

// sovraccarico

template <class tipoelem> void Lista<tipoelem>::operator =(const Lista<tipoelem>& l) //sovraccarico operatore di assegnamento

{

if (this!=&l) //non posso assegnare per esempio A=A;

{

svuota(); //svuoto

posizione ind=primolista();

posizione ind2=l.primolista();

while (!l.finelista(ind2))

{

// inslista(ind,l.leggilista(ind2)); //e inserisco in essa tutti gli elementi di l

ind=succlista(ind);

ind2=l.succlista(ind2);

} }

}

template <class tipoelem> bool Lista<tipoelem>::operator ==(Lista<tipoelem>& l) //sovraccarico 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 && (leggilista(ind)==l.leggilista(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) //sovraccarico output

{

Cella<tipoelem>\* ind=l.primolista();

while (!l.finelista(ind))

{

os<< l.leggilista(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 = succlista(p);

}

}

// fine modifiche zagaria

#endif

### 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 succlista (posizione)const;

posizione preclista (posizione)const;

void inslista (tipoelem,posizione &);

void canclista (posizione &);

private:

static const int max = 1024;

T \* elementi;

int lunghezza;

};

template <class T>Lista <T>::Lista()

{

crealista();

}

template <class T>Lista <T>::~Lista()

{

lunghezza = 0;

delete elementi;

}

template <class T>void Lista <T>::crealista()

{

lunghezza = 0;

elementi = new T[max];

}

template <class T>bool Lista <T>::listavuota() const

{

return (lunghezza == 0);

}

template <class T>typename Lista <T>::posizione Lista <T>::primolista() const

{

return 0;

}

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

{

if (0 <= p && p < lunghezza)

return (p + 1);

else

return p;

}

template <class T>typename Lista <T>::posizione Lista <T>::preclista(posizione p)const

{

if (0 < p && p < lunghezza)

return (p - 1);

else

return p;

}

template <class T>bool Lista <T>::finelista(posizione p) const

{

if (0 <= p && p <= lunghezza)

return (p == lunghezza);

else

return false;

}

template <class T>T Lista <T>::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

### 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\_ 12. #include "Cella\_L\_DP.h" 14. **template** <**class** tipoelem> 15. **class** Lista{ 16. **public**: 17. **typedef** Cella\_L\_DP<tipoelem>\* posizione; 19. Lista(); 20. Lista(**const** Lista&); 21. ~Lista(); 23. **void** crealista(); 25. **bool** listavuota() **const**; 26. posizione primolista() **const**; 27. posizione succlista(posizione) **const**; 28. posizione preclista(posizione) **const**; 29. **bool** finelista(posizione) **const**; 31. tipoelem leggilista(posizione) **const**; 33. **void** inslista(tipoelem, posizione&); 34. **void** scrivilista(tipoelem, posizione); 35. **void** canclista(posizione&); 37. **private**: 38. posizione lista; 39. }; 41. **template** <**class** tipoelem> 42. Lista<tipoelem>::Lista() 43. { 44. crealista(); 45. } 47. **template** <**class** tipoelem> 48. Lista<tipoelem>::Lista(**const** Lista& b) 49. { 50. crealista(); 51. **typename** Lista<tipoelem>::posizione ind = primolista(),ind2 = b.primolista(); 52. **while** (!b.finelista(ind2)) 53. { 54. inslista(b.leggilista(ind2),ind); 55. ind = succlista(ind); 56. ind2 = b.succlista(ind2); 57. } 58. } 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. } 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. } 81. **template** <**class** tipoelem> 82. **bool** Lista<tipoelem>::listavuota() **const** 83. { 84. **return** (lista->getSucc() == nullptr && lista->getPrec() == nullptr); 85. } 87. **template** <**class** tipoelem> 88. **typename** Lista<tipoelem>::posizione Lista<tipoelem>::primolista() **const** 89. { 90. **return** lista; 91. } 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. } 102. **template** <**class** tipoelem> 103. **typename** Lista<tipoelem>::posizione Lista<tipoelem>::preclista(posizione p) **const** 104. { 105. **return** p->getPrec(); 106. } 108. **template** <**class** tipoelem> 109. **bool** Lista<tipoelem>::finelista(posizione p) **const** 110. { 111. **return** (p->getSucc() == nullptr); 112. }  115. **template** <**class** tipoelem> 116. tipoelem Lista<tipoelem>::leggilista(posizione p) **const** 117. { 118. **return** p->getElem(); 119. } 121. **template** <**class** tipoelem> 122. **void** Lista<tipoelem>::scrivilista(tipoelem e, posizione p) 123. { 124. p->setElem(e); 125. } 127. **template** <**class** tipoelem> 128. **void** Lista<tipoelem>::inslista(tipoelem e, posizione& p) 129. { 130. **typename** Lista<tipoelem>::posizione temp; 132. temp = **new** Cella\_L\_DP<tipoelem>; 133. temp->setElem(e); 134. temp->setSucc(p); 135. temp->setPrec(p->getPrec()); 137. **if**(p == primolista()) 138. lista = temp; 139. **else** 140. p->getPrec()->setSucc(temp); 142. p->setPrec(temp); 144. p = temp; 145. }  148. **template** <**class** tipoelem> 149. **void** Lista<tipoelem>::canclista(posizione& p) 150. { 151. posizione temp; 152. temp = p; 153. **if**(p == primolista()) 154. { 155. **if**(p->getSucc() != nullptr) 156. { 157. lista = p->getSucc(); 158. lista->setPrec(nullptr); 159. } 160. } 161. **else** 162. { 163. preclista(p)->setSucc(p->getSucc()); 164. succlista(p)->setPrec(preclista(p)); 165. } 167. p = p->getSucc(); 168. **delete** temp; 169. } 171. #endif //LISTA\_H\_ |

#### 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\_ 8. **template** <**class** tipoelem> 9. **class** Cella\_L\_DP{ 10. **public**: 11. Cella\_L\_DP(); 12. ~Cella\_L\_DP(); 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. }; 26. **template** <**class** tipoelem> 27. Cella\_L\_DP<tipoelem>::Cella\_L\_DP() 28. { 29. succ = nullptr; 30. prec = nullptr; 31. } 33. **template** <**class** tipoelem> 34. Cella\_L\_DP<tipoelem>::~Cella\_L\_DP(){} 36. **template** <**class** tipoelem> 37. tipoelem Cella\_L\_DP<tipoelem>::getElem() **const** 38. { 39. **return** elem; 40. } 42. **template** <**class** tipoelem> 43. **void** Cella\_L\_DP<tipoelem>::setElem(tipoelem e) 44. { 45. elem = e; 46. } 48. **template** <**class** tipoelem> 49. Cella\_L\_DP<tipoelem>\* Cella\_L\_DP<tipoelem>::getSucc() **const** 50. { 51. **return** succ; 52. } 54. **template** <**class** tipoelem> 55. **void** Cella\_L\_DP<tipoelem>::setSucc(Cella\_L\_DP<tipoelem>\* e) 56. { 57. succ = e; 58. } 60. **template** <**class** tipoelem> 61. Cella\_L\_DP<tipoelem>\* Cella\_L\_DP<tipoelem>::getPrec() **const** 62. { 63. **return** prec; 64. } 66. **template** <**class** tipoelem> 67. **void** Cella\_L\_DP<tipoelem>::setPrec(Cella\_L\_DP<tipoelem>\* e) 68. { 69. prec = e; 70. } 72. #endif //CELLA\_L\_DP\_H\_ |

## Lista Ordinata

### Lista Ordinata con vettore:

#### 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>::leggilista(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) && leggilista(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

### Lista Ordinata con puntatori:

#### 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 succlista(posizione) const;
21. posizione preclista(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>::leggilista(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) && leggilista(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

## Pila

### Pila con puntatori:

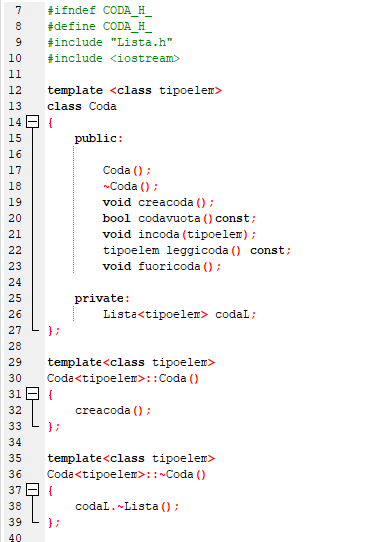
#### Pila.h

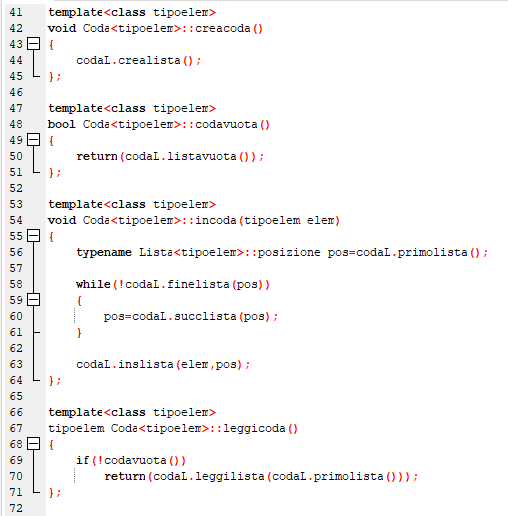
|  |
| --- |
| 1. /\*\* 2. \* @file Pila.h 3. \* Definizione della struttura dati "Pila". 4. \* @note Definizione di Pila per la realizzazione tramite puntatori. 5. \* @author Sconosciuto, modificato da Niccolo' Petti. 6. \* @date Anno Accademico 2018/19. 7. \*/ 8. #ifndef \_PILA\_H\_INCLUDED 9. #define \_PILA\_H\_INCLUDED 10. #include<assert.h> 11. #include "Cella\_Pila\_P\_Mono.h"  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 { 22. **public**: 23. **typedef** T tipoelem; 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(); 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**; 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**; 47. /\*\* 48. \* @brief Estrae 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(); 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); 61. **private**: 62. **typedef** Cella\_Pila\_P\_Mono<T> posizione; 63. posizione\* testa; 64. };  67. **template** <**class** T> Pila<T>::Pila() { 68. creapila(); 69. } 71. **template** <**class** T> Pila<T>::~Pila() { 72. **while**(!pilavuota()) 73. fuoripila(); 74. **delete** testa; 75. } 77. **template** <**class** T> **void** Pila<T>::creapila() { 78. testa = nullptr; 79. } 81. **template** <**class** T> **bool** Pila<T>::pilavuota() **const**{ 82. **return**(testa == nullptr); 83. } 85. **template** <**class** T> T Pila<T>::leggipila() **const**{ 86. assert(!pilavuota()); 87. **return** testa->leggiCella(); 88. } 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. } 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 |

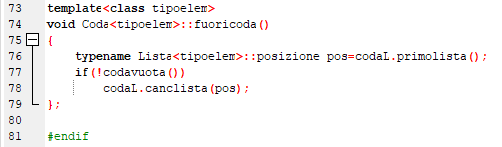
## Coda

Inserite tra le strutture intercambiabili le seguenti realizzazioni:

### Coda con lista







### Coda con puntatori

/\*

Coda.h

Created by: Mirco Sternativo

Edited by: Matteo Luceri

Date: 16-Mar-2015 // 30-Oct-2019

\*/

**#ifndef** \_CODA\_H

**#define** \_CODA\_H

**#include** "Cella.h"

**#include** <iostream>

**#include** <cstdlib>

**using** **namespace** std;

**template**<**class** **tipoelem**>

**class** Coda{

**public**:

//COSTRUTTORE E DISTRUTTORE ------------------------------------------------------------------

**Coda**(); //costruttore coda

**Coda**(Coda&); //costruttore di copia

**~Coda**(); //distruttore coda

//OPERATORI ----------------------------------------------------------------------------------

**void** **creacoda**();

**bool** **codavuota**() **const**;

**tipoelem** **leggicoda**() **const**;

**void** **incoda**(**tipoelem**);

**void** **fuoricoda**();

//NUOVI METODI

**void** **inverti\_coda**();

**void** **svuota**();

**private**:

//DEFINIZIONE DEI TIPI -----------------------------------------------------------------------

**typedef** Cella<**tipoelem**>\* posizione;

//VARIABILI ----------------------------------------------------------------------------------

posizione testa;

posizione fondo;

};

//IMPLEMENTAZIONI COSTRUTTORE E DISTRUTTORE ----------------------------------------------------------

**template**<**class** **tipoelem**>

**Coda<tipoelem>::Coda**(){

creacoda();

}

**template**<**class** **tipoelem**>

**Coda<tipoelem>::Coda**(Coda<**tipoelem**>& c){

//N.B.: questo costruttore effettua una copia o clone di un oggetto

creacoda();

**tipoelem** temp;

Coda<**tipoelem**> comodo;

**while** (!c.codavuota()){

comodo.incoda(c.leggicoda());

//copio gli elementi di c in una coda d'appoggio

c.fuoricoda(); //distruzione della coda

}

**while** (!comodo.codavuota()){

temp=comodo.leggicoda();

comodo.fuoricoda();

incoda(temp);

//copia degli elementi della coda d'appoggio nella nuova coda

c.incoda(temp);

//e ripristino c

}

}

**template**<**class** **tipoelem**>

**Coda<tipoelem>::~Coda**(){

**while** (!codavuota())

{

fuoricoda();

//eliminazione elementi coda

}

**delete** fondo;

//eliminazione riferimenti per inizio e fine coda

**delete** testa;

}

//IMPLEMENTAZIONI OPERATORI --------------------------------------------------------------------------

**template**<**class** **tipoelem**>

**void** **Coda<tipoelem>::creacoda**(){

testa=**nullptr**;

fondo=**nullptr**;

}

**template**<**class** **tipoelem**>

**bool** **Coda<tipoelem>::codavuota**() **const**{

**return** ((testa==**nullptr**));

//controllo esistenza coda tramite verifica puntatori inizio e fine

}

**template**<**class** **tipoelem**>

**tipoelem** **Coda<tipoelem>::leggicoda**() **const** {

**if** (!codavuota())

//precondizione coda non vuota

**return** (testa->leggicella());

//lettura elemento in testa

}

**template**<**class** **tipoelem**>

**void** **Coda<tipoelem>::incoda**(**tipoelem** a){

posizione temp=**new** Cella<**tipoelem**>;

//creazione nuovo elemento temp

temp->scrivicella(a);

//valorizzazione di temp

temp->scrivisucc(**nullptr**);

**if** (!codavuota())

fondo->scrivisucc(temp);

fondo=temp;

//se la coda non è vuota, l'elemento successivo è temp

**else**

testa=temp;

fondo=temp;

//altrimenti sta alla testa

}

**template**<**class** tipoelem>

**void** Coda<tipoelem>::**fuoricoda**(){

**if** (!codavuota()){

//precondizione coda non vuota

posizione temp=testa;

//puntatore all'elemento da eliminare

testa=testa->leggisucc();

**delete** temp;

//eliminazione elemento in testa

}

// NUOVI METODI--------------------------------------------------------------------------------------

**template**<**class** tipoelem>

**void** Coda<tipoelem>::inverti\_coda(){

tipoelem Elemento;

**if** (!codavuota()){

Elemento = leggicoda();

fuoricoda();

inverti\_coda();

incoda(Elemento);

}

}

**template**<**class** C>

**void** Coda<C>::svuota(){

**while** (!codavuota()){

fuoricoda();

}

};

**#endif** // CODA\_H

### 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++;

} }

## Coda di priorità

### 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) //sovraccarico 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**;

}

}

}

}

### Coda con priorità con lista ordinata

Lista con puntatori

Cella lista con puntatori

Lista ordinata   
tramite ereditarietà

Coda con priorità  
 tramite lista con vettore non ordinato

Elemento coda

Lista ordinata

Coda con priorità   
con lista non ordinata

Coda con priorità   
con lista ordinata e dinamica

Servizi coda con priorità

Servizi coda con priorità

Si fornisce uno schema riassuntivo del processo di creazione della struttura:

#### Cella\_Lista\_Con\_Puntatori\_Monodirezionale.h

/\*

Realizzazione: Cella per Lista con Puntatori Monodirezionale

Modificato da: Matteo Luceri

Autore : Sconosciuto

\*/

**#ifndef** CELLA\_LISTA\_CON\_PUNTATORI\_MONODIREZIONALE\_H\_

**#define** CELLA\_LISTA\_CON\_PUNTATORI\_MONODIREZIONALE\_H\_

**template**<**class** **T**>

**class** Cella\_LP\_Mono

{

**public**:

**typedef** **T** tipoelem;

**Cella\_LP\_Mono**();

**~Cella\_LP\_Mono**();

**void** **scriviCella**(tipoelem);

tipoelem **leggiCella**() **const**;

**void** **scriviSucc**(Cella\_LP\_Mono<**T**>\*);

Cella\_LP\_Mono<**T**>\* **leggiSucc**() **const**;

**private**:

tipoelem elemento;

Cella\_LP\_Mono<**T**>\* succ;

};

**template**<**class** **T**>

**Cella\_LP\_Mono<T>::Cella\_LP\_Mono**()

{

succ = **nullptr**;

}

**template**<**class** **T**>

**Cella\_LP\_Mono<T>::~Cella\_LP\_Mono**()

{

//dtor

}

**template**<**class** **T**>

**void** **Cella\_LP\_Mono<T>::scriviCella**(tipoelem label)

{

elemento = label;

}

**template**<**class** **T**>

**T** **Cella\_LP\_Mono<T>::leggiCella**() **const**

{

**return** elemento;

}

**template**<**class** **T**>

**void** **Cella\_LP\_Mono<T>::scriviSucc**(Cella\_LP\_Mono<**T**>\* c)

{

succ = c;

}

**template**<**class** **T**>

Cella\_LP\_Mono<**T**>\* **Cella\_LP\_Mono<T>::leggiSucc**() **const**

{

**return** succ;

}

**#endif** /\* CELLA\_LISTA\_CON\_PUNTATORI\_MONODIREZIONALE\_H\_ \*/

#### Lista\_Con\_Puntatori\_Monodirezionale.h

/\*

Realizzazione: Lista con Puntatori Monodirezionale

Modificato da Matteo Luceri

Autore : Sconosciuto

Nota Bene: Il costruttore di copia e' stato implementato in quanto

NECESSARIO per via di alcuni file del progetto in cui

vengono passate liste come parametri di funzioni e

restituite (dalle stesse) non per riferimento.

\*/

**#ifndef** LISTA\_CON\_PUNTATORI\_MONODIREZIONALE\_H\_

**#define** LISTA\_CON\_PUNTATORI\_MONODIREZIONALE\_H\_

**#include** "Cella\_Lista\_Con\_Puntatori\_Monodirezionale.h"

**template**<**class** **L**>

**class** Lista

{

**public**:

**typedef** Cella\_LP\_Mono<**L**>\* posizione;

**typedef** **L** tipoelem;

**Lista**();

**Lista**(**const** Lista&);

**~Lista**();

**void** **crealista**();

**bool** **listavuota**() **const**;

**bool** **finelista**(posizione) **const**;

posizione **primolista**() **const**;

posizione **succlista**(posizione) **const**;

posizione **preclista**(posizione) **const**;

tipoelem **leggilista**(posizione) **const**;

**void** **scrivilista**(tipoelem, posizione);

**void** **inslista**(tipoelem, posizione&);

**void** **canclista**(posizione &);

**private**:

posizione lista;

};

**template** <**class** **L**>

**Lista<L>::Lista**()

{

crealista();

}

**template** <**class** **L**>

**Lista<L>::Lista**(**const** Lista& b)

{

crealista();

**typename** Lista<**L**>::posizione ind = primolista(),ind2 = b.primolista();

**while** (!b.finelista(ind2))

{

inslista(b.leggilista(ind2),ind);

ind = succlista(ind);

ind2 = b.succlista(ind2);

}

}

**template** <**class** **L**>

**Lista<L>::~Lista**()

{

posizione p = primolista();

**while**(p->leggiSucc() != **nullptr**)

{

posizione temp = p;

p=p->leggiSucc();

**delete** temp;

}

}

**template** <**class** **L**>

**void** **Lista<L>::crealista**()

{

lista = **new** Cella\_LP\_Mono<**L**>;

lista->scriviSucc(**nullptr**);

}

**template** <**class** **L**>

**bool** **Lista<L>::listavuota**() **const**

{

**return** (lista->leggiSucc() == **nullptr**);

}

**template** <**class** **L**>

**bool** **Lista<L>::finelista**(posizione pos) **const**

{

**return** (pos->leggiSucc() == **nullptr**);

}

**template** <**class** **L**>

**typename** Lista<**L**>::posizione **Lista<L>::primolista**() **const**

{

**return** lista;

}

**template** <**class** **L**>

**typename** Lista<**L**>::posizione **Lista<L>::succlista**(posizione pos) **const**

{

**if** (lista->leggiSucc() != **nullptr**)

**return** (pos->leggiSucc());

**else**

**return** pos;

}

**template** <**class** **L**>

**typename** Lista<**L**>::posizione **Lista<L>::preclista**(posizione pos) **const**

{

**typename** Lista<**L**>::posizione temp = primolista();

**if**(pos == temp)

temp = **nullptr**;

**else**

{

**while**(succlista(temp) != pos)

temp = succlista(temp);

}

**return** temp;

}

**template** <**class** **L**>

**L** **Lista<L>::leggilista**(posizione pos) **const**

{

**return** (pos->leggiCella());

}

**template** <**class** **L**>

**void** **Lista<L>::scrivilista**(tipoelem elem, posizione pos)

{

pos->scriviCella(elem);

}

**template** <**class** **L**>

**void** **Lista<L>::inslista**(tipoelem elem, posizione &pos)

{

**typename** Lista<**L**>::posizione temp;

temp = **new** Cella\_LP\_Mono<**L**>;

temp->scriviCella(elem);

temp->scriviSucc(pos);

**if** (pos == primolista())

lista = temp;

**else**

preclista(pos)->scriviSucc(temp);

pos = temp;

}

**template** <**class** **L**>

**void** **Lista<L>::canclista**(posizione & pos)

{

**typename** Lista<**L**>::posizione temp = pos;

**if**(pos != lista)

preclista(pos)->scriviSucc(pos->leggiSucc());

**else**

lista = pos->leggiSucc();

pos = pos->leggiSucc();

**delete** temp;

}

**#endif** /\* LISTA\_CON\_PUNTATORI\_MONODIREZIONALE\_H\_ \*/

#### Lista\_Ordinata\_Ereditata.h

/\*

Definizione della struttura dati "Lista Ordinata".

Definizione di Lista Ordinata per la realizzazione tramite ereditarietà da Lista.

Autore: Andrea Esposito.

Modificata da: Matteo Luceri

\*/

**#ifndef** LISTA\_ORDINATA\_EREDITATA\_H\_

**#define** LISTA\_ORDINATA\_EREDITATA\_H\_

**#include** "Lista\_Con\_Puntatori\_Monodirezionale.h"

**template** <**class** **T**>

**class** ListaOrdinata : **private** Lista<**T**>

{

**public**:

**typedef** **typename** Lista<**T**>::posizione posizione;

**ListaOrdinata**();

**ListaOrdinata**(**const** ListaOrdinata&);

**~ListaOrdinata**();

**void** **crealista**();

**bool** **listavuota**() **const**;

**bool** **finelista**(posizione) **const**;

posizione **primolista**() **const**;

posizione **succlista**(posizione) **const**;

posizione **preclista**(posizione) **const**;

**T** **leggilista**(posizione) **const**;

**void** **inslista**(**T**);

**void** **canclista**(posizione&);

};

**template** <**class** **T**>

**ListaOrdinata<T>::ListaOrdinata**() :

Lista<**T**>() // call parent constructor

{

}

**template** <**class** **T**>

**ListaOrdinata<T>::ListaOrdinata**(**const** ListaOrdinata<**T**>& l):

Lista<**T**>(l) // call parent constructor

{

}

**template** <**class** **T**>

**ListaOrdinata<T>::~ListaOrdinata**()

{

Lista<**T**>::~Lista();

}

**template** <**class** **T**>

**void** **ListaOrdinata<T>::crealista**()

{

Lista<**T**>::crealista();

}

**template** <**class** **T**>

**bool** **ListaOrdinata<T>::listavuota**() **const**

{

**return** Lista<**T**>::listavuota();

}

**template** <**class** **T**>

**typename** ListaOrdinata<**T**>::posizione **ListaOrdinata<T>::primolista**() **const**

{

**return** Lista<**T**>::primolista();

}

**template** <**class** **T**>

**bool** **ListaOrdinata<T>::finelista**(posizione p) **const**

{

**return** Lista<**T**>::finelista(p);

}

**template** <**class** **T**>

**typename** ListaOrdinata<**T**>::posizione **ListaOrdinata<T>::succlista**(posizione p) **const**

{

**return** Lista<**T**>::succlista(p);

}

**template** <**class** **T**>

**typename** ListaOrdinata<**T**>::posizione **ListaOrdinata<T>::preclista**(posizione p) **const**

{

**return** Lista<**T**>::preclista(p);

}

**template** <**class** **T**>

**void** **ListaOrdinata<T>::inslista**(**T** el)

{

posizione p = primolista();

**if**(!listavuota())

**while**(leggilista(p) < el && !finelista(p))

p = succlista(p);

Lista<**T**>::inslista(el, p);

}

**template** <**class** **T**>

**void** **ListaOrdinata<T>::canclista**(posizione &p)

{

Lista<**T**>::canclista(p);

}

**template** <**class** **T**>

**T** **ListaOrdinata<T>::leggilista**(posizione p) **const**

{

**return** Lista<**T**>::leggilista(p);

}

**#endif** /\* LISTA\_ORDINATA\_EREDITATA\_H\_ \*/

#### Elemento\_Coda\_Con\_Priorita.h

/\*

Realizzazione: Tipo di elemento della coda con priorità

Modificato da: Matteo Luceri

Autore : Sconosciuto

\*/

**#ifndef** ELEMENTO\_CODA\_CON\_PRIORITA\_H\_

**#define** ELEMENTO\_CODA\_CON\_PRIORITA\_H\_

**#include** <iostream>

**#include** <cstdlib>

**using** **namespace** std;

**template**<**class** **X**> **class** Priorielem

{

**public**:

//dichiarazione di tipo

**typedef** **float** priorita; //la priorità è di tipo numerico (valore minore=priorità più alta)

**typedef** **X** tipoelem;

//costruttori

**Priorielem**();

**Priorielem**(**const** Priorielem&);

**Priorielem**(priorita,tipoelem);

//distruttore di default

//setter e getter

**void** **scrivipriorita**(priorita);

priorita **leggipriorita**() **const**;

**void** **scrivielem**(tipoelem);

tipoelem **leggielem**() **const**;

//sovraccarichi

**void** **operator=**(Priorielem);

**bool** **operator==**(Priorielem);

**bool** **operator<**(Priorielem);

**bool** **operator>**(Priorielem);

**private**:

priorita prior; //priorità

tipoelem elem; //informazione

};

**template** <**class** **X**> **Priorielem<X>::Priorielem**() //costruttore generico

{

prior=9999;

}

**template** <**class** **X**> **Priorielem<X>::Priorielem**(**const** Priorielem& p) //costruttore di copia

{

prior=p.leggipriorita();

elem=p.leggielem();

}

**template** <**class** **X**> **Priorielem<X>::Priorielem**(priorita p,tipoelem e) //costruttore specifico

{

prior=p;

elem=e;

}

**template** <**class** **X**> **void** **Priorielem<X>::scrivipriorita**(priorita p)

{

prior=p;

}

**template** <**class** **X**> **void** **Priorielem<X>::scrivielem**(tipoelem e)

{

elem=e;

}

**template** <**class** **X**> **float** **Priorielem<X>::leggipriorita**() **const**

{

**return**(prior);

}

**template** <**class** **X**> **X** **Priorielem<X>::leggielem**() **const**

{

**return**(elem);

}

//sovraccarichi

**template** <**class** **X**> **void** **Priorielem<X>::operator=**(Priorielem<**X**> p) //assegnamento

{

prior=p.leggipriorita();

elem=p.leggielem();

}

**template** <**class** **X**> **bool** **Priorielem<X>::operator==**(Priorielem<**X**> p) //uguaglianza

{

**return** (elem==p.leggielem());

}

**template** <**class** **X**> **bool** **Priorielem<X>::operator<**(Priorielem<**X**> p) //maggioranza (solo sulla priorità)

{

**return** (prior<p.leggipriorita());

}

**template** <**class** **X**> **bool** **Priorielem<X>::operator>**(Priorielem<**X**> p) //minoranza (solo sulla priorità)

{

**return** (prior>p.leggipriorita());

}

//sovraccarico output

**template**<**class** **X**> ostream& **operator<<**(ostream& os, **const** Priorielem<**X**>& p)

{

os<<"("<<p.leggipriorita()<<"|"<<p.leggielem()<<")";

**return**(os);

}

**#endif** /\* ELEMENTO\_CODA\_CON\_PRIORITA\_H\_ \*/

#### Coda\_Con\_Priorita.h

/\*

Realizzazione: Coda con priorità tramite una lista ordinata

Note: Valore di priorità minore indica priorità maggiore

Modificato da: Matteo Luceri

Autore : Sconosciuto

\*/

**#ifndef** CODA\_CON\_PRIORITA\_H\_

**#define** CODA\_CON\_PRIORITA\_H\_

**#include** "Lista\_Ordinata\_Ereditata.h"

**#include** <iostream>

**#include** <cstdlib>

**#include** "Elemento\_Coda\_Con\_Priorita.h"

**using** **namespace** std;

//realizzazione di una

**template**<**class** **P**> **class** Prioricoda

{

**public**:

//definizione di tipo

**typedef** Priorielem<**P**> tipoelem;

//costruttori

**Prioricoda**();

**Prioricoda**(**const** Prioricoda<**P**>&);

//distruttore di default

//operatori di specifica

**void** **creaprioricoda**();

**void** **inserisci**(tipoelem);

tipoelem **min**() **const**;

**void** **cancellamin**();

//operatori ereditati dell'insieme

**bool** **insiemevuoto**() **const**;

**bool** **appartiene**(tipoelem) **const**;

**private**:

ListaOrdinata<tipoelem> prioricoda; //la coda con priorità di fatto è una lista

**friend** ostream& **operator<<** (ostream& o, **const** Prioricoda<**P**>& p) //sovraccarico output

{

o<<p.prioricoda;

**return** o;

}

};

**template**<**class** **P**> **Prioricoda<P>::Prioricoda**() //costruttore generico

{

creaprioricoda();

}

**template** <**class** **P**> **Prioricoda<P>::Prioricoda**(**const** Prioricoda<**P**>& p) //costruttore di copia

{

creaprioricoda();

prioricoda=p.prioricoda;

}

**template**<**class** **P**> **void** **Prioricoda<P>::creaprioricoda**() //crea la coda con priorità

{

prioricoda.crealista();

}

**template**<**class** **P**> **void** **Prioricoda<P>::inserisci**(tipoelem a) //inserimento

{

prioricoda.inslista(a);

}

**template**<**class** **P**> Priorielem<**P**> **Prioricoda<P>::min**() **const** //restituisce il minimo della coda

{

tipoelem m;

**if** (!prioricoda.listavuota()) //precondizione coda non vuota

{

**typename** ListaOrdinata<tipoelem>::posizione indice=prioricoda.primolista();

m= prioricoda.leggilista(indice);

}

**return** m;

}

**template**<**class** **P**> **void** **Prioricoda<P>::cancellamin**() //elimina il minimo dalla coda

{

**if** (!prioricoda.listavuota()) //precondizione coda non vuota

{

**typename** ListaOrdinata<tipoelem>::posizione indice=prioricoda.primolista();

prioricoda.canclista(indice);

}

}

**template**<**class** **P**> **bool** **Prioricoda<P>::insiemevuoto**() **const** //verifica se la coda è vuota

{

**return** (prioricoda.listavuota());

}

**template**<**class** **P**> **bool** **Prioricoda<P>::appartiene**(tipoelem a) **const** //verifica se l'elemento appartiene alla coda

{

**bool** trovato=**false**;

**if** (!prioricoda.listavuota())

{

**typename** ListaOrdinata<tipoelem>::posizione indice=prioricoda.primolista(); //ricerca

**while** (!prioricoda.finelista(indice) && !trovato)

{

**if** (prioricoda.leggilista(indice)==a) //se l'elemento corrente è il cercato

trovato=**true**;

**else** indice=prioricoda.succlista(indice);

}

}

**return**(trovato);

}

**#endif** /\* CODA\_CON\_PRIORITA\_H\_ \*/

#### Servizi\_Coda\_Con\_Priorita.h

/\*

Definizione dei servizi per la struttura Coda Con Priorità

Autore: Sconosciuto

Modificata da: Matteo Luceri

\*/

**#ifndef** SERVIZI\_CODA\_CON\_PRIORITA\_H\_

**#define** SERVIZI\_CODA\_CON\_PRIORITA\_H\_

**#include** <string>

**#include** <iostream>

**#include** <fstream>

**#include** <exception>

**#include** "Coda\_Con\_Priorita.h"

**template** <**class** **T**>

**void** **stampaPrioriCoda**(Prioricoda<**T**>&);

**template** <**class** **T**>

**void** **inputPrioriCodaDaFile**(Prioricoda<**T**>&, std::ifstream&);

**template** <**class** **T**>

**void** **outputPrioriCodaSuFile**(Prioricoda<**T**>&, std::ofstream&);

// Implementazione

**template** <**class** **T**>

**void** **stampaPrioriCoda**(Prioricoda<**T**>& p)

{

**if**(!p.insiemevuoto())

{

**typename** Prioricoda<**T**>::tipoelem el = p.min();

std::cout << el.leggielem() << " - con priorita: " << el.leggipriorita() << std::endl;

p.cancellamin();

stampaPrioriCoda(p);

p.inserisci(el);

}

}

**template** <**class** **T**>

**void** **inputPrioriCodaDaFile**(Prioricoda<**T**>& p, std::ifstream& file)

{

**if**(!file.fail())

{

**if**(file.peek() != ifstream::traits\_type::*eof*())

**while**(!file.eof())

{

**T** content;

string s;

**getline**(file, s,',');

file >> content;

**typename** Prioricoda<**T**>::tipoelem::priorita priori = (**typename** Prioricoda<**T**>::tipoelem::priorita) **atof**(s.c\_str());

p.inserisci(**typename** Prioricoda<**T**>::tipoelem(priori, content));

}

}

**else**

**throw** std::runtime\_error("Errore di apertura del file");

}

**template** <**class** **T**>

**void** **outputPrioriCodaSuFile**(Prioricoda<**T**>& p, std::ofstream& file)

{

**if**(!file.fail())

{

**if**(!p.insiemevuoto())

{

**typename** Prioricoda<**T**>::tipoelem el = p.min();

file << el.leggipriorita() << "," << el.leggielem() << std::endl;

p.cancellamin();

outputPrioriCodaSuFile(p, file);

p.inserisci(el);

}

}

**else**

**throw** std::runtime\_error("Errore di apertura del file");

}

**#endif** /\* SERVIZI\_CODA\_CON\_PRIORITA\_H\_ \*/

## Albero n-ario

### 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{ 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. //COSTRUTTORE 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), FALSO ALTRIMENTI 38. **bool** foglia(nodo) **const**; 39. //METODO CHE RESTITUISCE IL NODO PRIMOFIGLIO DI UN NODO PASSATO 40. nodo primofiglio(nodo) **const**; 41. //METODO CHE RESTITUISCE VERO SE IL NODO PASSATO NON HA FRATELLI SUCCESSIVI, FALSO ALTRIMENTI 42. **bool** ultimofratello(nodo) **const**; 43. //METODO CHE RESTITUISCE IL NODO FRATELLO SUCCESSIVO DEL NODO PASSATO 44. nodo succfratello(nodo)**const**; 45. //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO COME NODO PRIMOFIGLIO DEL NODO PASSATO 46. **void** insprimosottoalbero(nodo, Albero&); 47. //METODO CHE INSERISCE LA RADICE, E TUTTI I SUOI DISCENDENTI, DELL'ALBERO PASSATO COME UN NODO FRATELLO SUCCESSIVO DEL NODO PASSATO 48. **void** inssottoalbero(nodo, Albero&); 49. //METODO CHE PASSATO UN NODO CANCELLA LO STESSO E TUTTI I SUOI DISCENDENTI 50. **void** cancsottoalbero(nodo); 51. //METODO CHE SCRIVE L'ETICHETTA DI TIPO TIPOELEM ALL'INTERNO DEL NODO PASSATO 52. **void** scrivinodo(nodo&, tipoelem); 53. //METODO CHE RESTITUISCE L'ETICHETTA DEL NODO PASSATO 54. tipoelem legginodo(nodo) **const**; 56. **private**: 58. BinAlbero<tipoelem> tree; 59. //METODO CHE COPIA UN ALBERO PASSATO PARTENDO DAL NODO SINISTRO 60. **void** copia\_albero\_sx(**const** BinAlbero<tipoelem>& ,**const** nodo& ,nodo); 61. //METODO CHE COPIA L'ALBERO PASSATO PARTENDO DAL NODO DESTRO 62. **void** copia\_albero\_dx(**const** BinAlbero<tipoelem>& ,**const** nodo& ,nodo); 63. };  66. **template** <**class** tipoelem> 67. Albero<tipoelem>::Albero(){ 68. creaalbero(); 69. } 71. **template** <**class** tipoelem> 72. Albero<tipoelem>::Albero(**const** Albero& a){ 73. **if**(!a.tree.binalberovuoto()) 74. { 75. tree.insbinradice(); 76. nodo n = tree.binradice(); 77. nodo secondo\_albero = a.tree.binradice(); 78. tree.scrivinodo(n, a.tree.legginodo(secondo\_albero)); 80. **if**(a.tree.sinistrovuoto(a.tree.binradice())) 81. { 82. **if**(!a.tree.destrovuoto(a.tree.binradice())) 83. copia\_albero\_dx(a.tree,a.tree.figliodestro(a.tree.binradice()),tree.binradice()); 84. } 85. **else** 86. copia\_albero\_sx(a.tree,a.tree.figliosinistro(a.tree.binradice()),tree.binradice()); 87. } 88. } 90. **template** <**class** tipoelem> 91. Albero<tipoelem>::~Albero(){ 92. //RICHIAMA AUTOMATICAMENTE IL DISTRUTTORE 93. //DELL'ALBERO BINARIO 94. } 96. **template** <**class** tipoelem> 97. **void** Albero<tipoelem>::creaalbero(){ 98. //RICHIAMA AUTOMATICAMENTE IL COSTRUTTORE 99. //DELL'ALBERO BINARIO 100. }  103. **template** <**class** tipoelem> 104. **bool** Albero<tipoelem>::alberovuoto() **const**{ 105. **return** tree.binalberovuoto(); 106. } 108. **template** <**class** tipoelem> 109. **void** Albero<tipoelem>::insradice(){ 110. tree.insbinradice(); 111. } 113. **template** <**class** tipoelem> 114. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::radice() **const**{ 115. **return** tree.binradice(); 116. } 118. **template** <**class** tipoelem> 119. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::padre(**typename** Albero<tipoelem>::nodo n){ 120. **return** tree.binpadre(n); 121. } 123. **template** <**class** tipoelem> 124. **bool** Albero<tipoelem>::foglia(**typename** Albero<tipoelem>::nodo n) **const**{ 125. **if**(tree.sinistrovuoto(n)) 126. **return** **true**; 127. **else** 128. **return** **false**; 129. } 131. **template** <**class** tipoelem> 132. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::primofiglio(**typename** Albero<tipoelem>::nodo n) **const**{ 133. **return** tree.figliosinistro(n); 134. } 136. **template** <**class** tipoelem> 137. **bool** Albero<tipoelem>::ultimofratello(**typename** Albero<tipoelem>::nodo n) **const**{ 138. **if**(tree.destrovuoto(n)) 139. **return** **true**; 140. **else** 141. **return** **false**; 142. } 144. **template** <**class** tipoelem> 145. **typename** Albero<tipoelem>::nodo Albero<tipoelem>::succfratello(**typename** Albero<tipoelem>::nodo n)**const**{ 146. **return** tree.figliodestro(n); 147. } 149. **template**<**class** tipoelem> 150. **void** Albero<tipoelem>::copia\_albero\_sx(**const** BinAlbero<tipoelem>& other, **const** nodo& radice\_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)); 156. **if**(!other.sinistrovuoto(radice\_sottoalbero)) 157. copia\_albero\_sx(other, other.figliosinistro(radice\_sottoalbero), sx); 158. **if**(!other.destrovuoto(radice\_sottoalbero)) 159. copia\_albero\_dx(other, other.figliodestro(radice\_sottoalbero), sx); 160. } 162. **template** <**class** tipoelem> 163. **void** Albero<tipoelem>::copia\_albero\_dx(**const** BinAlbero<tipoelem>& other, **const** nodo& radice\_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)); 169. **if**(!other.sinistrovuoto(radice\_sottoalbero)) 170. copia\_albero\_sx(other, other.figliosinistro(radice\_sottoalbero), dx); 171. **if**(!other.destrovuoto(radice\_sottoalbero)) 172. copia\_albero\_dx(other, other.figliodestro(radice\_sottoalbero), dx); 173. } 175. **template** <**class** tipoelem> 176. **void** Albero<tipoelem>::insprimosottoalbero(**typename** Albero<tipoelem>::nodo n, Albero& a){ 177. nodo temp; 178. **if**(tree.sinistrovuoto(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))); 187. **if**(tree.sinistrovuoto(tree.figliosinistro(n))){ 188. **if**(!tree.destrovuoto(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. } 196. tree.cancsottobinalbero(tree.figliosinistro(n)); 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. }  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 fratello successivo 208. di n. n non è la radice\*/ 210. nodo temp; 211. **if**(n!=tree.binradice()){ 212. **if**(tree.destrovuoto(n)){ 213. copia\_albero\_dx(a.tree,a.radice(),n); 214. } 215. **else**{ 216. Albero<tipoelem> T; 217. T.insradice(); 219. temp=T.radice(); 220. T.scrivinodo(temp,tree.legginodo(tree.figliodestro(n))); 222. **if**(tree.sinistrovuoto(tree.figliodestro(n))){ 223. **if**(!tree.destrovuoto(tree.figliodestro(n))){ 224. copia\_albero\_dx(tree,tree.figliodestro(tree.figliodestro(n)),temp); 225. } 226. } 227. **else**{ 228. copia\_albero\_sx(tree,tree.figliosinistro(tree.figliodestro(n)),temp); 229. } 231. cancsottoalbero(tree.figliodestro(n)); 233. copia\_albero\_dx(a.tree,a.radice(),n); 234. temp=tree.figliodestro(n); 235. copia\_albero\_dx(T.tree,T.radice(),temp); 237. } 238. } 240. } 242. **template** <**class** tipoelem> 243. **void** Albero<tipoelem>::cancsottoalbero(**typename** Albero<tipoelem>::nodo n){ 244. /\* L'albero è ottenuto togliendo il sottoalbero di radice n e tutti i suoi discendenti\*/ 245. tree.cancsottobinalbero(n); 246. } 248. **template** <**class** tipoelem> 249. **void** Albero<tipoelem>::scrivinodo(**typename** Albero<tipoelem>::nodo& n, tipoelem elem){ 250. tree.scrivinodo(n,elem); 251. } 253. **template** <**class** tipoelem> 254. tipoelem Albero<tipoelem>::legginodo(**typename** Albero<tipoelem>::nodo n) **const**{ 255. **return** tree.legginodo(n); 256. } 258. #endif // ALBERONARIO\_H |

#### BinAlbero: realizzazione totalmente dinamico

|  |
| --- |
| 1. #ifndef BINALBERI\_H\_INCLUDED 2. #define BINALBERI\_H\_INCLUDED 3. #include "Nodo\_Albero\_Binario.h" 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{ 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, 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 figliodestro(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** cancsottobinalbero(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** insfigliodestro(Nodo); 55. **private**: 56. Nodo radice; 57. };  60. **template** <**class** TIPOETICHETTA> 61. BinAlbero<TIPOETICHETTA>::BinAlbero(){ 62. creabinalbero(); 63. } 65. **template** <**class** TIPOETICHETTA> 66. BinAlbero<TIPOETICHETTA>::~BinAlbero(){ 67. /\* if(radice!=nil) 68. cancsottobinalbero(binradice()); 69. \*/ 71. } 73. **template** <**class** TIPOETICHETTA> 74. **void** BinAlbero<TIPOETICHETTA>::creabinalbero(){ 75. radice=nil; 76. } 78. **template** <**class** TIPOETICHETTA> 79. **bool** BinAlbero<TIPOETICHETTA>::binalberovuoto()**const**{ 80. **if**(radice==nil) 81. **return** **true**; 82. **else** 83. **return** **false**; 84. } 86. **template** <**class** TIPOETICHETTA> 87. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binradice()**const**{ 88. **return** radice; 89. } 91. **template** <**class** TIPOETICHETTA> 92. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::binpadre(Nodo padre)**const**{ 93. **return** padre->getPadre(); 94. } 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. } 104. **template** <**class** TIPOETICHETTA> 105. **bool** BinAlbero<TIPOETICHETTA>::destrovuoto(Nodo dx)**const**{ 106. **if**(dx->getFiglioDx()==nil) 107. **return** **true**; 108. **else** 109. **return** **false**; 110. } 112. **template** <**class** TIPOETICHETTA> 113. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliosinistro(Nodo sx)**const**{ 114. **return** sx->getFiglioSx(); 115. } 117. **template** <**class** TIPOETICHETTA> 118. **typename** BinAlbero<TIPOETICHETTA>::Nodo BinAlbero<TIPOETICHETTA>::figliodestro(Nodo dx)**const**{ 119. **return** dx->getFiglioDx(); 120. } 122. **template** <**class** TIPOETICHETTA> 123. **void** BinAlbero<TIPOETICHETTA>::costrbinalbero(BinAlbero<TIPOETICHETTA> &A,BinAlbero<TIPOETICHETTA> &B){ 124. radice=**new** Nodo; 125. radice->setPadre(nil); 127. **if**(!A.binalberovuoto()){ 128. radice->setFiglioSx(A.binradice()); 129. radice->getFiglioSx()->setPadre(radice); 130. } 131. **else** 132. radice->setFiglioSx(nil);  135. **if**(!B.binalberovuoto()){ 136. radice->setFiglioDx(B.binradice()); 137. radice->getFiglioDx()->setPadre(radice); 138. } 139. **else** 140. radice->setFiglioDx(nil); 141. } 143. **template** <**class** TIPOETICHETTA> 144. **void** BinAlbero<TIPOETICHETTA>::cancsottobinalbero(Nodo r){ 146. **if**(!sinistrovuoto(r)) 147. cancsottobinalbero(r->getFiglioSx()); 149. **if**(!destrovuoto(r)) 150. cancsottobinalbero(r->getFiglioDx()); 152. **if**(radice!=r) 153. { 154. Nodo temp; 155. temp=binpadre(r); 157. **if**(temp->getFiglioSx()==r) 158. temp->setFiglioSx(nil); 159. **else** 160. temp->setFiglioDx(nil); 161. } 162. **else** 163. radice=nil; 165. **delete** r; 167. } 169. **template** <**class** TIPOETICHETTA> 170. TIPOETICHETTA BinAlbero<TIPOETICHETTA>::legginodo(Nodo n)**const**{ 171. **return** n->getEtichetta(); 172. } 174. **template** <**class** TIPOETICHETTA> 175. **void** BinAlbero<TIPOETICHETTA>::scrivinodo(Nodo n, TIPOETICHETTA e){ 176. n->setEtichetta(e); 177. } 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. } 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. } 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. } 205. #endif // BINALBERI\_H\_INCLUDED |

#### Nodo\_Albero\_Binario

|  |
| --- |
| 1. #ifndef NODO\_ALBERO\_BINARIO\_H\_INCLUDED 2. #define NODO\_ALBERO\_BINARIO\_H\_INCLUDED 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. };  47. **template** <**class** TIPOETICHETTA> 48. Cella\_Binalbero<TIPOETICHETTA>::Cella\_Binalbero(){ 49. FiglioDx=nullptr; 50. FiglioSx=nullptr; 51. Padre=nullptr; 52. } 54. **template** <**class** TIPOETICHETTA> 55. Cella\_Binalbero<TIPOETICHETTA>::~Cella\_Binalbero(){ 56. } 58. **template** <**class** TIPOETICHETTA> 59. **void** Cella\_Binalbero<TIPOETICHETTA>::setFiglioSx(Cella\_Binalbero<TIPOETICHETTA>\* sx){ 60. FiglioSx=sx; 61. } 63. **template** <**class** TIPOETICHETTA> 64. **void** Cella\_Binalbero<TIPOETICHETTA>::setFiglioDx(Cella\_Binalbero<TIPOETICHETTA>\* dx){ 65. FiglioDx=dx; 66. } 68. **template** <**class** TIPOETICHETTA> 69. **void** Cella\_Binalbero<TIPOETICHETTA>::setPadre(Cella\_Binalbero<TIPOETICHETTA>\* p){ 70. Padre=p; 71. } 73. **template** <**class** TIPOETICHETTA> 74. **void** Cella\_Binalbero<TIPOETICHETTA>::setEtichetta(TIPOETICHETTA& e){ 75. etichetta=e; 76. } 78. **template** <**class** TIPOETICHETTA> 79. Cella\_Binalbero<TIPOETICHETTA>\* Cella\_Binalbero<TIPOETICHETTA>::getFiglioSx(){ 80. **return** FiglioSx; 81. } 83. **template** <**class** TIPOETICHETTA> 84. Cella\_Binalbero<TIPOETICHETTA>\* Cella\_Binalbero<TIPOETICHETTA>::getFiglioDx(){ 85. **return** FiglioDx; 86. } 88. **template** <**class** TIPOETICHETTA> 89. Cella\_Binalbero<TIPOETICHETTA>\* Cella\_Binalbero<TIPOETICHETTA>::getPadre(){ 90. **return** Padre; 91. } 93. **template** <**class** TIPOETICHETTA> 94. TIPOETICHETTA Cella\_Binalbero<TIPOETICHETTA>::getEtichetta(){ 95. **return** etichetta; 96. } 98. **template** <**class** TIPOETICHETTA> 99. **bool** Cella\_Binalbero<TIPOETICHETTA>::operator == (Cella\_Binalbero<TIPOETICHETTA>  b){ 100. **if**(getEtichetta()==b.getEtichetta()) 101. **return** **true**; 102. **else** 103. **return** **false**; 104. } 105. **template** <**class** TIPOETICHETTA> 106. Cella\_Binalbero<TIPOETICHETTA>::Cella\_Binalbero(**const** Cella\_Binalbero& c){ 107. etichetta=c.getEtichetta(); 108. FiglioDx=c.getFiglioDx(); 109. FiglioSx=c.getFiglioSx(); 110. Padre=c.getPadre(); 111. } 113. **template** <**class** TIPOETICHETTA> 114. **void** Cella\_Binalbero<TIPOETICHETTA>::operator=(**const** Cella\_Binalbero& c ){ 115. etichetta=c.getEtichetta(); 116. Padre=c.getPadre(); 117. FiglioDx=c.getFiglioDx(); 118. FiglioSx=c.getFiglioSx(); 119. } 120. #endif // NODO\_ALBERO\_BINARIO\_H\_INCLUDED |

### AlberoNario: realizzazione primofiglio/fratello

|  |
| --- |
| 1. #ifndef ALBERO\_N\_H 2. #define ALBERO\_N\_H 3. #include "Cella\_albero\_n.h" 4. #include <iostream> 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. { 14. **public**: 15. Albero(); //costruttore 16. **virtual** ~Albero();//distruttore 17. **static** constexpr Cella<T>\* nil=nullptr; 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** cancsottoalbero(nodo);//metodo che cancella un sotto albero 33. **void** scrivinodo(nodo, tipoelem);//metodo che scrive nell'etichetta del nodo 34. tipoelem legginodo(nodo);//funzione che ritorna il contenuto dell'etichetta del nodo 35. **private**: 36. nodo root;//nodo radice 37. }; 39. **template** <**class** T> Albero<T>::Albero() 40. { 41. creaalbero(); 42. } 44. **template** <**class** T> Albero<T>::~Albero() {} 46. **template** <**class** T> **void** Albero<T>::creaalbero() 47. { 48. root=NULL; 49. } 51. **template** <**class** T> **bool** Albero<T>::alberovuoto() **const** 52. { 53. **return**(**this**->root==NULL); 54. } 56. **template** <**class** T> **void** Albero<T>::insradice() 57. { 58. **if**(alberovuoto()) 59. { 60. root= **new** Cella<T>; 61. root->setpadre(NULL); 62. root->setprimofiglio(NULL); 63. root->setfratellosucc(NULL); 64. } 66. } 68. **template** <**class** T> 69. Cella<T>\* Albero<T>::radice() **const**{ 70. **return** (root); 71. } 73. **template** <**class** T> 74. Cella<T>\* Albero<T>::padre(nodo n){ 75. **if**(n!=radice()){ 76. **return** (n->getpadre()); 77. } 78. } 80. **template** <**class** T> 81. **bool** Albero<T>::foglia(nodo n){ 82. **return** (n->getprimofiglio()==NULL); 83. } 85. **template** <**class** T> 86. Cella<T>\* Albero<T>::primofiglio(nodo n){ 87. **return** (n->getprimofiglio()); 89. } 91. **template** <**class** T> 92. **bool** Albero<T>::ultimofratello(nodo n){ 93. **return** (n->getfratellosucc()==NULL); 94. } 96. **template** <**class** T> 97. Cella<T>\* Albero<T>::succfratello(nodo n){ 98. **if**(!(ultimofratello(n))){ 99. **return** (n->getfratellosucc()); 100. } 101. } 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. } 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. } 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. } 139. **delete** n; 140. } 142. **template** <**class** T> 143. **void** Albero<T>::scrivinodo(nodo n, tipoelem a){ 144. n->setetichetta(a); 145. } 147. **template** <**class** T> 148. T Albero<T>::legginodo(nodo n){ 149. **return** (n->getetichetta()); 150. } 152. #endif // ALBERO\_N\_H |

#### Cella\_albero\_n

|  |
| --- |
| 1. #ifndef CELLA\_H 2. #define CELLA\_H 3. #include <iostream> 4. **using** **namespace** std; 5. **template** <**class** T> 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. }; 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. } 39. **template** <**class** T> Cella<T>::~Cella() {} 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. } 49. **template** <**class** T> **void** Cella<T>::setetichetta(tipoelem a) 50. { 51. **this**->etichetta=a; 52. } 54. **template** <**class** T> T Cella<T>::getetichetta() 55. { 56. **return** etichetta; 57. } 59. **template** <**class** T> **void** Cella<T>::setprimofiglio(Cella\* a) 60. { 61. **this**->primofiglio=a; 62. } 64. **template** <**class** T> Cella<T>\* Cella<T>::getprimofiglio() 65. { 66. **return** primofiglio; 67. } 69. **template** <**class** T> **void** Cella<T>::setfratellosucc(Cella\* a) 70. { 71. **this**->fratellosucc=a; 72. } 74. **template** <**class** T> Cella<T>\* Cella<T>::getfratellosucc() 75. { 76. **return** fratellosucc; 77. } 79. **template** <**class** T> **void** Cella<T>::setpadre(Cella\* a) 80. { 81. **this**->padre=a; 82. } 84. **template** <**class** T> Cella<T>\* Cella<T>::getpadre() 85. { 86. **return** padre; 87. } 89. **template** <**class** T> **bool** Cella<T>::operator==(Cella a) 90. { 91. **return** (**this**->etichetta==a.getetichetta()); 92. } 94. #endif // CELLA\_H |

## Insiemi

### Insiemi con lista

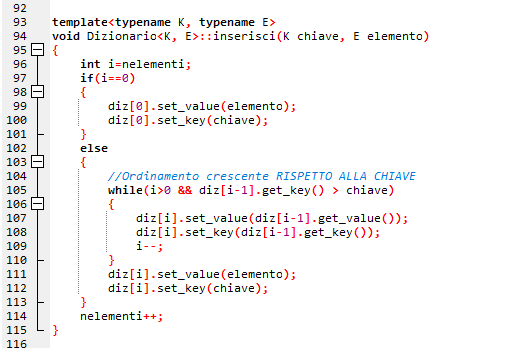
|  |
| --- |
| 1. /\*\* 2. \* @file Insieme.h 3. \* Definizione della struttura dati "Insieme". 4. \* @note Definizione di Insieme per la realizzazione tramite lista. 5. \* @author Sconosciuto, modificato da Andrea Esposito. 6. \* @date Anno Accademico 2018/19. 7. \*/ 8. #ifndef INSIEME\_H\_ 9. #define INSIEME\_H\_ 11. #include "Lista.h" 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(); 27. /\*\* 28. \* @brief Crea un insieme vuoto. 29. \* @post Il nuovo insieme i è l'insieme vuoto (i={}). 30. \*/ 31. **void** creainsieme(); 33. /\*\* 34. \* @brief Controlla se l'insieme è vuoto. 35. \* @return true se l'insieme è vuoto, false altrimenti. 36. \*/ 37. **bool** insiemevuoto() **const**; 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**; 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); 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); 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 conterrè tutti e soli gli elementi 65. \* iniziali e gli elementi dell'insieme i. 66. \*/ 67. **void** unione(**const** Insieme& i); 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 conterrè tutti e soli gli elementi 74. \* presenti sia nell'insieme iniziale che in i. 75. \*/ 76. **void** intersezione(**const** Insieme& i); 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 conterrè tutti e soli gli elementi 83. \* iniziali che non appartengono anche a i. 84. \*/ 85. **void** differenza(**const** Insieme& i); 87. **private**: 88. Lista<TipoElem> setlist; 89. }; 91. **template** <**class** TipoElem> 92. Insieme<TipoElem>::Insieme() 93. { 94. creainsieme(); 95. } 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. } 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. } 118. **template** <**class** TipoElem> 119. **void** Insieme<TipoElem>::creainsieme() 120. { 121. setlist.crealista(); 122. } 124. **template** <**class** TipoElem> 125. **bool** Insieme<TipoElem>::insiemevuoto() **const** 126. { 127. **return** setlist.listavuota(); 128. } 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. } 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. } 155. **template** <**class** TipoElem> 156. **void** Insieme<TipoElem>::cancella(**const** TipoElem& e) 157. { 158. **typename** Lista<TipoElem>::posizione p = setlist.primolista(); 159. **while** (!setlist.finelista(p) && setlist.leggilista(p) != e) 160. p = setlist.succlista(p); 161. **if** (setlist.leggilista(p) == e) 162. setlist.canclista(p); 163. } 165. **template** <**class** TipoElem> 166. **void** Insieme<TipoElem>::unione(**const** Insieme<TipoElem>& i) 167. { 168. **typename** Lista<TipoElem>::posizione p = i.setlist.primolista(); 169. **while** (!i.setlist.finelista(p)) 170. { 171. inserisci(i.setlist.leggilista(p)); 172. p = i.setlist.succlista(p); 173. } 174. } 176. **template** <**class** TipoElem> 177. **void** Insieme<TipoElem>::intersezione(**const** Insieme<TipoElem>& i) 178. { 179. **typename** Lista<TipoElem>::posizione p = setlist.primolista(); 181. **while**(!setlist.finelista(p)) 182. { 183. **if**(!i.appartiene(setlist.leggilista(p))) 184. cancella(setlist.leggilista(p)); 185. **else** 186. p = setlist.succlista(p); 187. } 188. } 190. **template** <**class** TipoElem> 191. **void** Insieme<TipoElem>::differenza(**const** Insieme<TipoElem>& i) 192. { 193. **typename** Lista<TipoElem>::posizione p = i.setlist.primolista(); 195. **while**(!setlist.finelista(p)) 196. { 197. **if**(i.appartiene(setlist.leggilista(p))) 198. cancella(setlist.leggilista(p)); 199. **else** 200. p = setlist.succlista(p); 201. } 202. } 204. #endif /\* INSIEME\_H\_ \*/ |

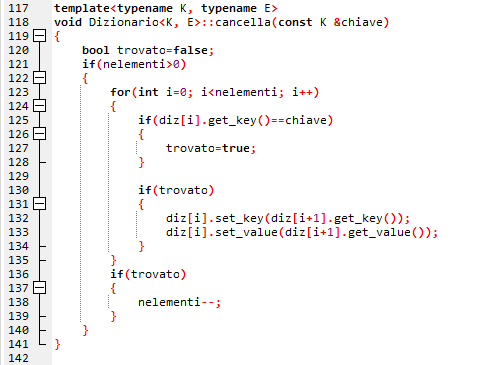
## Dizionario

### Dizionario con vettore ordinato

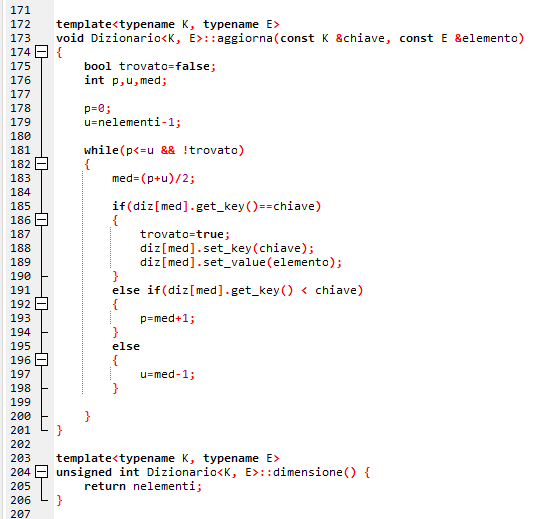
# 

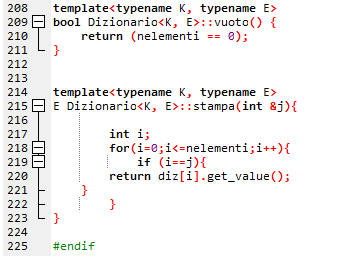












### Dizionario con lista

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. template<typename K, typename E>
27. Dizionario<K, E>::Dizionario() {
28. crea();
29. }
30. template<typename K, typename E>
31. Dizionario<K, E>::~Dizionario() {}
32. template<typename K, typename E>
33. void Dizionario<K, E>::crea(){
34. table.crealista();
35. }
36. template<typename K, typename E>
37. bool Dizionario<K, E>::appartiene(const K &chiave){
38. bool trovato=false;
39. typename Lista<Entry<K, E> >::posizione pos=table.primolista();
40. while(!trovato && !table.finelista(pos)){
41. if(table.leggilista(pos).get\_key() == chiave){
42. trovato=true;
43. }
44. else{
45. pos=table.succlista(pos);
46. }
47. }
48. return trovato;
49. }
50. template<typename K, typename E>
51. void Dizionario<K, E>::inserisci(K chiave, E elemento){
52. typename Lista<Entry<K, E> >::posizione pos=table.primolista();
53. Entry<K,E> elem;
54. elem.set\_key(chiave);
55. elem.set\_value(elemento);
56. table.inslista(elem,pos);
57. }
58. template<typename K, typename E>
59. void Dizionario<K, E>::cancella(const K &chiave){
60. typename Lista<Entry<K, E> >::posizione pos=table.primolista();
61. bool trovato=false;
62. while(!trovato && !table.finelista(pos)){
63. if(table.leggilista(pos).get\_key() == chiave){
64. trovato=true;
65. table.canclista(pos);
66. }
67. else{
68. pos=table.succlista(pos);
69. }
70. }
71. }
72. template<typename K, typename E>
73. E Dizionario<K, E>::recupera(const K &chiave){
74. typename Lista<Entry<K, E> >::posizione pos=table.primolista();
75. Entry<K,E> elem;
76. E valore;
77. bool trovato = false;
78. while(!table.finelista(pos) && !trovato){
79. if(table.leggilista(pos).get\_key() == chiave){
80. valore=table.leggilista(pos).get\_value();
81. elem.set\_value(valore);
82. trovato=true;
83. }
84. pos=table.succlista(pos);
85. }
86. return elem.get\_value();
87. }
88. template<typename K, typename E>
89. void Dizionario<K, E>::aggiorna(const K &chiave, E &elemento){
90. typename Lista<Entry<K, E> >::posizione pos=table.primolista();
91. Entry<K,E> elem;
92. bool trovato = false;
93. while(!table.finelista(pos) && !trovato){
94. if(table.leggilista(pos).get\_key() == chiave){
95. elem.set\_key(chiave);
96. elem.set\_value(elemento);
97. table.scrivilista(elem,pos);
98. trovato=true;
99. }
100. pos=table.succlista(pos);
101. }
102. }
103. #endif // \_DIZIONARIO\_H

## Grafi

### Matrice di Adiacenza

|  |
| --- |
| 1. /\*\* 2. \* @file Grafo.h 3. \* Definizione struttura dati Grafo realizzata mediante Matrice di Adiacenza 4. \* @author Graziano Montanaro. 5. \* @date Anno Accademico 2018/19. 6. \*/ 8. #ifndef GRAFO\_H\_INCLUDED 9. #define GRAFO\_H\_INCLUDED 11. #include "CellaGrafo.h" 12. #include "CellaNodo.h" 13. #include <iostream> 14. #include "Lista.h" 15. #define maxNodi 1024  18. **template** <**class** tipoElem, **class** tipoPeso> 19. **class** Grafo 20. { 21. **public**: 22. **class** nodo; 23. Grafo(); 24. ~Grafo(); 25. **void** creagrafo(); 26. **bool** grafovuoto(); 27. **void** insarco(nodo&,nodo&); 28. **void** cancarco(nodo&,nodo&); 29. **void** insnodo(nodo &); 30. **void** cancnodo(nodo&); 31. Lista<nodo> adiacenti(nodo&); 32. **bool** esistenodo(nodo&); 33. **bool** esistearco(nodo&,nodo&); 34. **void** scrivinodo(tipoElem,nodo&); 35. **void** scriviarco(tipoPeso,nodo&,nodo&); 36. tipoElem legginodo(nodo&); 37. tipoPeso leggiarco(nodo&,nodo&);  40. **private**: 42. unsigned **int** nodiUsati; 43. unsigned **int** primoLibero(); 44. CellaGrafo<tipoPeso> matrice[maxNodi][maxNodi]; 45. CellaNodo<tipoElem> nodi[maxNodi]; 46. };  49. // Definizione del tipo nodo 51. **template** <**class** tipoElem, **class** tipoPeso> 52. **class** Grafo<tipoElem,tipoPeso>::nodo 53. { 54. **public**: 55. nodo(); 56. nodo(**const** nodo&); 57. ~nodo(); 58. **int** legginodo(); 59. **void** scrivinodo(**int** n); 60. **bool**  operator == ( **const** nodo& ); 61. **bool**  operator != ( **const** nodo& );  64. **private**: 65. **int** valore; 66. };   70. **template** <**class** tipoElem, **class** tipoPeso> 71. Grafo<tipoElem , tipoPeso>::Grafo() 72. { 73. nodiUsati = 0; 74. **for**(**int** i=0;i<maxNodi;i++) 75. { 76. nodi[i].setPresente(**false**); 77. **for**(**int** j=0;i<maxNodi;i++) 78. matrice[i][j].setUsato(**false**); 79. } 80. } 82. **template** <**class** tipoElem, **class** tipoPeso> 83. Grafo<tipoElem , tipoPeso>::~Grafo() 84. { 86. } 88. **template** <**class** tipoElem, **class** tipoPeso> 89. **bool** Grafo<tipoElem , tipoPeso>::grafovuoto() 90. { 91. **return** nodiUsati == 0; 92. } 94. **template** <**class** tipoElem, **class** tipoPeso> 95. **bool** Grafo<tipoElem, tipoPeso>::esistenodo(nodo &n) 96. { 97. **if**(n.legginodo()>=0 && n.legginodo()<nodiUsati) 98. **return** nodi[n.legginodo()].getPresente(); 99. **else** 100. **return** **false**; 101. } 103. **template** <**class** tipoElem, **class** tipoPeso> 104. tipoPeso Grafo<tipoElem , tipoPeso>::leggiarco(nodo& n1, nodo& n2) 105. { 106. **if**(esistearco(n1,n2)) 107. **return** matrice[n1.legginodo()][n2.legginodo()].getPeso();  110. } 112. **template** <**class** tipoElem, **class** tipoPeso> 113. **void** Grafo<tipoElem , tipoPeso>::insnodo(nodo & n) 114. { 115. **if**(!esistenodo(n)) 116. { 117. **if**(nodiUsati<maxNodi) 118. { 119. unsigned **int**  i = primoLibero(); 120. nodi[i].setPresente(**true**); 121. nodi[i].setArchi(0); 122. n.scrivinodo(i); 123. nodiUsati++; 124. } 125. } 126. } 128. **template** <**class** tipoElem, **class** tipoPeso> 129. **void** Grafo<tipoElem , tipoPeso>::cancnodo(nodo& n) 130. { 131. **if**(esistenodo(n)) 132. { 133. nodo\* temp = **new** nodo; 134. **for**(**int** i=0;i<nodiUsati;i++) 135. { 136. temp->scrivinodo(i); 137. **if**(esistearco(\*temp,n)) 138. cancarco(\*temp,n); 139. **if**(esistearco(n,\*temp)) 140. cancarco(n,\*temp); 141. } 142. nodi[n.legginodo()].setPresente(**false**); 143. nodiUsati--;  146. } 148. } 150. **template** <**class** tipoElem, **class** tipoPeso> 151. **void** Grafo<tipoElem , tipoPeso>::insarco(nodo &n1, nodo &n2) 152. { 153. **if**(!esistearco(n1,n2)) 154. { 155. matrice[n1.legginodo()][n2.legginodo()].setUsato(**true**); 156. nodi[n1.legginodo()].setArchi(nodi[n1.legginodo()].getArchi()+1); 157. nodi[n2.legginodo()].setArchi(nodi[n2.legginodo()].getArchi()+1); 158. }  161. } 163. **template** <**class** tipoElem, **class** tipoPeso> 164. **void** Grafo<tipoElem , tipoPeso>::cancarco(nodo& n1, nodo& n2) 165. { 166. **if**(esistearco(n1,n2)) 167. { 168. matrice[n1.legginodo()][n2.legginodo()].setUsato(**false**); 169. nodi[n1.legginodo()].setArchi(nodi[n2.legginodo()].getArchi() -1); 170. nodi[n2.legginodo()].setArchi(nodi[n2.legginodo()].getArchi() -1); 172. } 173. } 175. **template** <**class** tipoElem, **class** tipoPeso> 176. **void** Grafo<tipoElem , tipoPeso>::scrivinodo(tipoElem e, nodo& n) 177. { 178. **if**(esistenodo(n)) 179. { 180. nodi[n.legginodo()].setEtichetta(e); 181. } 183. } 185. **template** <**class** tipoElem, **class** tipoPeso> 186. tipoElem Grafo<tipoElem , tipoPeso>::legginodo(nodo& n) 187. { 188. **if**(esistenodo(n)) 189. { 190. **return** nodi[n.legginodo()].getEtichetta(); 191. } 192. } 194. **template** <**class** tipoElem, **class** tipoPeso> 195. **void** Grafo<tipoElem , tipoPeso>::scriviarco(tipoPeso p,nodo& n1, nodo& n2) 196. { 197. **if**(!esistearco(n1,n2)) 198. { 199. matrice[n1.legginodo()][n2.legginodo()].setPeso(p); 200. } 202. } 204. **template** <**class** tipoElem, **class** tipoPeso> 205. **bool** Grafo<tipoElem , tipoPeso>::esistearco(nodo& n1, nodo& n2) 206. { 207. **if**(esistenodo(n1)) 208. **if**(esistenodo(n2)) 209. **return** matrice[n1.legginodo()][n2.legginodo()].getUsato(); 211. }  214. **template** <**class** tipoElem, **class** tipoPeso> 215. Lista<**typename** Grafo<tipoElem,tipoPeso>::nodo> Grafo<tipoElem , tipoPeso>::adiacenti(nodo& n) 216. { 217. Lista<nodo> l; 218. **typename** Lista<nodo>::posizione p = l.primolista(); 219. **for**(**int** i=0;i<nodiUsati;i++) 220. { 221. **if**(matrice[n.legginodo()][i].getUsato()) 222. { 223. nodo\* temp = **new** nodo(); 224. temp->scrivinodo(i); 225. l.inslista(\*temp,p); 226. } 227. } 228. // Nodi di cui n è coda 229. /\*for(int i=0;i<nodiUsati;i++) 230. { 231. if(matrice[i][n].getUsato()) 232. if(!matrice[n][i].getUsato()) 233. l.inslista(i,p); 234. }\*/ 235. **return** l; 236. } 238. **template** <**class** tipoElem, **class** tipoPeso> 239. unsigned **int** Grafo<tipoElem,tipoPeso>::primoLibero() 240. { 241. nodo i; 242. i.scrivinodo(-1); 243. **bool** libero=**false**; 244. **while**(!libero) 245. { 246. i.scrivinodo(i.legginodo()+1); 247. **if**(nodi[i.legginodo()].getPresente()==**false**) 248. libero = **true**; 249. } 250. **return** i.legginodo(); 251. } 253. //Operatori NODO 255. **template** <**class** tipoElem, **class** tipoPeso> 256. Grafo<tipoElem,tipoPeso>::nodo::nodo() 257. { 258. valore = -1; 259. }   263. **template** <**class** tipoElem, **class** tipoPeso> 264. Grafo<tipoElem,tipoPeso>::nodo::nodo(**const** nodo& n) 265. { 266. valore = n.valore; 267. }   271. **template** <**class** tipoElem, **class** tipoPeso> 272. Grafo<tipoElem,tipoPeso>::nodo::~nodo() 273. { 275. }  278. **template** <**class** tipoElem, **class** tipoPeso> 279. **int** Grafo<tipoElem,tipoPeso>::nodo::legginodo() 280. { 281. **return** valore; 282. }  285. **template** <**class** tipoElem, **class** tipoPeso> 286. **void** Grafo<tipoElem,tipoPeso>::nodo::scrivinodo(**int** n) 287. { 288. valore = n; 289. } 291. **template** <**class** tipoElem, **class** tipoPeso> 292. **bool** Grafo<tipoElem,tipoPeso>::nodo:: operator == ( **const** nodo& n) 293. { 294. **return** valore == n.valore; 295. } 297. **template** <**class** tipoElem, **class** tipoPeso> 298. **bool** Grafo<tipoElem,tipoPeso>::nodo:: operator != ( **const** nodo& n) 299. { 300. **return** valore != n.valore; 301. }  304. #endif // GRAFO\_H\_INCLUDED |

#### CellaGrafo.h

|  |
| --- |
| 1. /\*\* 2. \* @file CellaGrafo.h 3. \* Definizione della Cella presente nella matrice di Grafo.h ( matrice di adiacenza) 4. \* @author Graziano Montanaro. 5. \* @date Anno Accademico 2018/19. 6. \*/  9. #ifndef CELLAGRAFO\_H\_INCLUDED 10. #define CELLAGRAFO\_H\_INCLUDED  13. **template** <**class** tipoPeso> 14. **class** CellaGrafo 15. { 16. **public**: 17. CellaGrafo(); 18. CellaGrafo(**const** CellaGrafo<tipoPeso>& ); 19. ~CellaGrafo(); 20. **void** setUsato(**bool**); 21. **void** setPeso(tipoPeso); 22. **bool** getUsato(); 23. tipoPeso getPeso(); 24. **bool** operator == ( **const** CellaGrafo<tipoPeso> & ); 25. **bool** operator != ( **const** CellaGrafo<tipoPeso> &); 26. **void** operator = ( **const** CellaGrafo<tipoPeso> & ); 27. **private**: 28. **bool** usato; 29. tipoPeso peso; 30. }; 32. **template** <**class** tipoPeso> 33. CellaGrafo<tipoPeso>::CellaGrafo() 34. { 35. usato = **false**; 36. peso = tipoPeso(); 37. } 39. **template** <**class** tipoPeso> 40. CellaGrafo<tipoPeso>::CellaGrafo(**const** CellaGrafo<tipoPeso>& c) 41. { 42. usato = c.usato; 43. peso = c.peso; 44. } 46. **template** <**class** tipoPeso> 47. CellaGrafo<tipoPeso>::~CellaGrafo() 48. { 50. } 52. **template** <**class** tipoPeso> 53. **void** CellaGrafo<tipoPeso>::setUsato(**bool** b) 54. { 55. usato = b; 56. } 58. **template** <**class** tipoPeso> 59. **void** CellaGrafo<tipoPeso>::setPeso(tipoPeso p) 60. { 61. peso = p; 62. } 64. **template** <**class** tipoPeso> 65. **bool** CellaGrafo<tipoPeso>::getUsato() 66. { 67. **return** usato; 68. } 70. **template** <**class** tipoPeso> 71. tipoPeso CellaGrafo<tipoPeso>::getPeso() 72. { 73. **return** peso; 74. } 76. **template** <**class** tipoPeso> 77. **bool** CellaGrafo<tipoPeso>::operator == (**const** CellaGrafo<tipoPeso> & c) 78. { 79. **return** peso == c.peso; 80. } 82. **template** <**class** tipoPeso> 83. **bool** CellaGrafo<tipoPeso>::operator != (**const** CellaGrafo<tipoPeso> & c) 84. { 85. **return** peso != c.peso; 86. } 88. **template** <**class** tipoPeso> 89. **void** CellaGrafo<tipoPeso>::operator = (**const** CellaGrafo<tipoPeso> & c) 90. { 91. peso = c.peso; 92. usato = c.usato; 93. } 95. #endif // CELLAGRAFO\_H\_INCLUDED |

#### CellaNodo.h

|  |
| --- |
| 1. /\*\* 2. \* @file CellaNodo.h 3. \* Definizione della Cella presente nel vettore di Nodi di Grafo.h ( matrice di adiacenza) 4. \* @author Graziano Montanaro. 5. \* @date Anno Accademico 2018/19. 6. \*/ 8. #ifndef CellaNodo\_H\_INCLUDED 9. #define CellaNodo\_H\_INCLUDED 11. **template** <**class** tipoElem> 12. **class** CellaNodo 13. { 14. **public**: 15. CellaNodo(); 16. CellaNodo(**const** CellaNodo<tipoElem>&); 17. ~CellaNodo(); 18. **void** setPresente(**bool**); 19. **void** setEtichetta(tipoElem ); 20. **void** setArchi(**int**); 21. **void** setIndice(unsigned **int**); 22. unsigned **int** getIndice(); 23. **bool** getPresente(); 24. tipoElem getEtichetta(); 25. **int** getArchi(); 26. **bool** operator == (**const** CellaNodo<tipoElem>&); 27. **bool** operator != (**const** CellaNodo<tipoElem>&); 28. **void** operator = (**const** CellaNodo<tipoElem>&); 30. **private**: 31. **bool** presente; 32. tipoElem etichetta; 33. **int** archi; 34. unsigned **int** indice; 35. }; 37. **template** <**class** tipoElem> 38. CellaNodo<tipoElem>::CellaNodo() 39. { 40. presente = **false**; 41. etichetta = tipoElem(); 42. archi = 0; 43. indice = 0; 44. } 46. **template** <**class** tipoElem> 47. CellaNodo<tipoElem>::CellaNodo(**const** CellaNodo<tipoElem>& n) 48. { 49. presente = n.presente; 50. etichetta = n.etichetta; 51. archi = n.archi; 52. indice = n.indice; 53. } 55. **template** <**class** tipoElem> 56. CellaNodo<tipoElem>::~CellaNodo() 57. { 59. }  62. **template** <**class** tipoElem> 63. **void** CellaNodo<tipoElem>::setPresente(**bool** b) 64. { 65. presente = b; 66. } 68. **template** <**class** tipoElem> 69. **void** CellaNodo<tipoElem>::setEtichetta(tipoElem e) 70. { 71. etichetta = e; 72. } 74. **template** <**class** tipoElem> 75. **void** CellaNodo<tipoElem>::setArchi(**int** a) 76. { 77. archi = a; 78. } 80. **template** <**class** tipoElem> 81. **void** CellaNodo<tipoElem>::setIndice(unsigned **int** a) 82. { 83. indice = a; 84. } 86. **template** <**class** tipoElem> 87. unsigned **int** CellaNodo<tipoElem>::getIndice() 88. { 89. **return** indice; 90. } 92. **template** <**class** tipoElem> 93. tipoElem CellaNodo<tipoElem>::getEtichetta() 94. { 95. **return** etichetta; 96. } 98. **template** <**class** tipoElem> 99. **bool** CellaNodo<tipoElem>::getPresente() 100. { 101. **return** presente; 102. }   106. **template** <**class** tipoElem> 107. **int** CellaNodo<tipoElem>::getArchi() 108. { 109. **return** archi; 110. } 112. **template** <**class** tipoElem> 113. **bool** CellaNodo<tipoElem>:: operator ==(**const** CellaNodo<tipoElem>& n) 114. { 115. **return** etichetta == n.etichetta; 116. } 118. **template** <**class** tipoElem> 119. **bool** CellaNodo<tipoElem>:: operator !=(**const** CellaNodo<tipoElem>& n) 120. { 121. **return** etichetta != n.etichetta; 122. } 124. **template** <**class** tipoElem> 125. **void** CellaNodo<tipoElem>:: operator = (**const** CellaNodo<tipoElem>& n) 126. { 127. presente = n.presente; 128. etichetta = n.etichetta; 129. archi = n.archi; 130. indice = n.indice; 131. } 133. #endif // CellaNodo\_H\_INCLUDED |

### 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. \*/
8. #ifndef \_GRAFO\_H
9. #define \_GRAFO\_H
11. #include <iostream>
12. #include <cstdlib>
13. #include "Lista.h"
14. #include "Adiacenza.h"
16. //definizione della classe grafo (orientato, etichettato e pesato)
17. //realizzazione tramite vettore (di lunghezza maxnodi) con liste (monodirezionali dinamiche) di adiacenza
19. **template**<**class** tipoElem,**class** tipoPeso> **class** Grafo
20. {
21. **public**:
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)
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;

36. Grafo(); // costruttore
37. ~Grafo(); // distruttore

40. unsigned **int** n\_nodi();
42. //operatori di specifica
43. **void** creagrafo();
44. **bool** grafovuoto() **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);
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")


65. };
67. //n viene passato nel crea grafo
68. **template**<**class** tipoElem,**class** tipoPeso> Grafo<tipoElem,tipoPeso>::Grafo() //costruttore specifico
69. {
70. creagrafo();
71. }
73. **template**<**class** tipoElem,**class** tipoPeso> Grafo<tipoElem,tipoPeso>::~Grafo() //distruttore
74. {
76. }
78. **template**<**class** tipoElem,**class** tipoPeso>
79. **void** Grafo<tipoElem,tipoPeso>::creagrafo() //crea il grafo
80. {
81. maxnodi = 1000;
82. nelementi=0; //dico che ci sono 0 nodi
83. **for** (**int** i=0; i<1000; i++) //inizializzo il vettore mettendo a false l'appartenenza di tutti nodi
84. {
85. table[i].esiste=**false**;
86. }
87. }
89. **template**<**class** tipoElem,**class** tipoPeso>
90. **bool** Grafo<tipoElem,tipoPeso>::grafovuoto() **const** //restituisce true se il grafo è vuoto, false altrimenti
91. {
92. **return** (nelementi==0);
93. }
95. **template**<**class** tipoElem,**class** tipoPeso>
96. **void** Grafo<tipoElem,tipoPeso>::insnodo(nodo &n) //inserisce il nodo che sarà identificato dall'indice n
97. {
98. //passaggio del parametro per indirizzo perchè la variabile sarà modificata
99. **if** (!esistenodo(n) && nelementi<1000) // precondizione nodo non appartenente
100. {
101. n=primolibero(); //la posizione in cui metterò il nodo del vettore sarà la prima libera
102. table[n].esiste=**true**; //setto a true il suo campo esiste
103. nelementi++; //aumento il contatore di nodi del grafo
104. }
105. }
107. **template**<**class** tipoElem,**class** tipoPeso>
108. **void** Grafo<tipoElem,tipoPeso>::insarco(nodo n,nodo m) //inserisce l'arco che esce dal nodo n ed entra in m
109. {
110. **if** (esistenodo(n) && esistenodo(m) && !esistearco(n,m))//precondizione nodi appartenenti e arco non esistente
111. {
112. **typename** Lista<adiacente>::posizione indice=table[n].adiac.primolista();
113. adiacente temp; //creo l'elemento da inserire
114. temp.scrivinodo(m); //setto l'adiacente
115. table[n].adiac.inslista(temp,indice); //inserisco il nodo m negli adiacenti di n (in prima posizione)
116. }
117. }
119. **template**<**class** tipoElem,**class** tipoPeso>
120. **void** Grafo<tipoElem,tipoPeso>::cancnodo(nodo n) //elimina il nodo n
121. {
122. **if** (esistenodo(n))// 1)---------------------------+
123. {
124. //|
125. **if** (table[n].adiac.listavuota())// 2)         //|
126. {
127. //|
128. **bool** libero=**true**;                           //|
129. **int** i=0;                                    //|
130. **while** (i<maxnodi && libero)                 //|
131. {
132. //|--- precondizione : nodo esistente 1),che non ha archi uscenti 2) nè entranti 3)
133. **if** (esistenodo(i))                        //|
134. {
135. //|
136. libero=!esistearco(i,n);                //|
137. }                                         //|
138. i++;                                      //|
139. }                                           //|
140. **if** (libero)// 3)------------------------------+
141. {
142. table[n].esiste=**false**; //adesso quel nodo non esiste più
143. table[n].etichetta=NULL; //svuoto l'etichetta
144. nelementi--; //e ho un nodo in meno nel grafo
145. }
146. }
147. }
148. }
150. **template**<**class** tipoElem,**class** tipoPeso>
151. **void** Grafo<tipoElem,tipoPeso>::cancarco(nodo n,nodo m)  //elimina l'arco che esce da n ed entra in m
152. {
153. **if** (esistenodo(n) && esistenodo(m) && esistearco(n,m))//precondizione nodi appartenenti e arco esistente
154. {
155. **bool** cancellato=**false**;
156. **typename** Lista<adiacente>::posizione indice=table[n].adiac.primolista();
157. **while** (!table[n].adiac.finelista(indice) && !cancellato) //scandisco la lista finchè non cancello l'elemento
158. {
159. **if** (table[n].adiac.leggilista(indice).legginodo()==m) //se trovo l'elemento lo cancello
160. {
161. table[n].adiac.canclista(indice); //elimino l'elemento
162. cancellato=**true**;
163. }
164. **else** indice=table[n].adiac.succlista(indice);
165. }
166. }
167. }
169. **template**<**class** tipoElem,**class** tipoPeso>
170. Lista<**typename** Grafo<tipoElem,tipoPeso>::nodo> Grafo<tipoElem,tipoPeso>::adiacenti(nodo n) **const** //restituisce la lista di adiacenti di n
171. {
172. Lista<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 restituitre (lista di nodo)
178. **while** (!table[n].adiac.finelista(indice)) //scansione della prima lista
179. {
180. temp=table[n].adiac.leggilista(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. }
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. }
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.finelista(indice) && !esiste) //scandisco la lista finchè non trovo l'elemento o è finita
204. {
205. **if** (table[n].adiac.leggilista(indice).legginodo()==m) esiste=**true**;
206. indice=table[n].adiac.succlista(indice);
207. }
208. }
209. **return** (esiste);
210. }
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. }
219. **template**<**class** tipoElem,**class** tipoPeso>
220. tipoElem Grafo<tipoElem,tipoPeso>::legginodo(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. }
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 appartenenente 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**;                                                      //quindi si fa un'unica scansione in cui si ricerca e si aggiorna)
235. adiacente temp(m,val);
236. **typename** Lista<adiacente>::posizione indice=table[n].adiac.primolista();
237. **while** (!table[n].adiac.finelista(indice) && !aggiornato) //scandisco la lista finchè non trovo l'elemento o è finita
238. {
239. **if** (table[n].adiac.leggilista(indice).legginodo()==m)
240. {
241. table[n].adiac.scrivilista(temp,indice);
242. aggiornato=**true**;
243. }
244. indice=table[n].adiac.succlista(indice);
245. }
246. }
247. }
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 vedere
254. {
255. //se esiste e poi si farebbe la seconda scansione per la lettura                                                                 //quindi si fa un'unica scansione in cui si ricerca e si legge)
256. **bool** letto=**false**;
257. **typename** Lista<adiacente>::posizione indice=table[n].adiac.primolista();
258. **while** (!table[n].adiac.finelista(indice) && !letto) //scandisco la lista finchè non trovo l'elemento o è finita
259. {
260. **if** (table[n].adiac.leggilista(indice).legginodo()==m)
261. {
262. a=table[n].adiac.leggilista(indice).leggipeso();
263. letto=**true**;
264. }
265. indice=table[n].adiac.succlista(indice);
266. }
267. }
268. **return**(a);
269. }

272. //operatori ausiliare
274. **template**<**class** tipoElem,**class** tipoPeso>
275. unsigned **int** Grafo<tipoElem,tipoPeso>::primolibero() **const** //restituisce la prima posizione 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 questo metodo viene chiamato solo se il vettore non è pieno
282. i++;
283. libero=!table[i].esiste;
284. }
285. **return** (i);
286. }

289. **template**<**class** tipoElem,**class** tipoPeso> unsigned **int** Grafo<tipoElem,tipoPeso>::n\_nodi()
290. {
291. **return** nelementi;
292. }

295. #endif

#### Adiacenza.h

|  |
| --- |
| 1. #ifndef \_ADIACENZA\_H 2. #define \_ADIACENZA\_H 4. #include <iostream> 5. #include <cstdlib> 7. **using** **namespace** std; 9. //tipo dell'elemento che sarà usato nella lista di adiacenza del grafo pesato 11. **template**<**class** nodo,**class** tipoPeso> **class** Adiacenza 12. { 13. **public**:  16. //costruttori (generico di default) 17. Adiacenza(); 18. Adiacenza(nodo,tipoPeso); 19. ~Adiacenza(); 20. //distruttore di default 22. //setter e getter 23. **void** scrivinodo(nodo); 24. nodo legginodo() **const**; 25. **void** scrivipeso(tipoPeso); 26. tipoPeso leggipeso() **const**;  29. **private**: 30. nodo adiacente; //riferimento al nodo adiacente 31. tipoPeso peso; //peso dell'arco 33. };  36. **template** <**class** nodo,**class** tipoPeso> Adiacenza<nodo,tipoPeso>::Adiacenza() //costruttore generico 37. { 38. } 40. **template** <**class** nodo,**class** tipoPeso> Adiacenza<nodo,tipoPeso>::Adiacenza(nodo n,tipoPeso p) //costruttore specifico 41. { 42. adiacente=n; 43. peso=p; 44. } 46. **template** <**class** nodo,**class** tipoPeso> Adiacenza<nodo,tipoPeso>::~Adiacenza() 47. { 48. //dtor 49. } 51. **template** <**class** nodo,**class** tipoPeso> **void** Adiacenza<nodo,tipoPeso>::scrivinodo(nodo n) 52. { 53. adiacente=n; 54. } 56. **template** <**class** nodo,**class** tipoPeso> **void** Adiacenza<nodo,tipoPeso>::scrivipeso(tipoPeso p) 57. { 58. peso=p; 59. } 61. **template** <**class** nodo,**class** tipoPeso> nodo Adiacenza<nodo,tipoPeso>::legginodo() **const** 62. { 63. **return**(adiacente); 64. } 66. **template** <**class** nodo,**class** tipoPeso> tipoPeso Adiacenza<nodo,tipoPeso>::leggipeso() **const** 67. { 68. **return**(peso); 69. }  72. //sovraccarico output 74. **template**<**class** nodo,**class** tipoPeso> ostream& operator<<(ostream& os, **const** Adiacenza<nodo,tipoPeso>& a) 75. { 76. os<<"("<<a.legginodo()<<"|"<<a.leggipeso()<<")"; 77. **return**(os); 78. } 80. #endif |