

Corso di Laurea in *Informatica e Comunicazione Digitale   
sede di Taranto*

CORSO DI   
*Algoritmi e Strutture Dati*

*Anno Accademico 2019/2020*

Docente: Studente:

Prof. Stefano Ferilli Matteo Luceri

Matr.: 677663

Sommario

[**1.** **Traccia** 3](#_Toc23673646)

[**2.** **Analisi** 3](#_Toc23673647)

[**2.1** **Analisi preliminare** 3](#_Toc23673648)

[**2.2** **Analisi integrativa** 4](#_Toc23673649)

[**3.** **Progettazione** 5](#_Toc23673650)

[**3.1** **Zaino e Valigia** 5](#_Toc23673651)

[**4.** **Implementazione** 9](#_Toc23673673)

[**4.** **Realizzazione** 12](#_Toc23673709)

[**4.1.1** **Astro.h** 12](#_Toc23673710)

[**4.1.2** **Astro.cpp** 12](#_Toc23673711)

[**4.1.3** **Gioco.h** 19](#_Toc23673712)

[**4.1.4** **Gioco.cpp** 20](#_Toc23673713)

[**4.1.5** **Mappa.h** 36](#_Toc23673716)

[**4.1.6** **Interfaccia.cpp** 36](#_Toc23673717)

[**4.1.7** **Oggetti.h** 37](#_Toc23673718)

[**4.1.8** **Oggetti.cpp** 37](#_Toc23673719)

[**4.1.9** **Oggetto.h** 39](#_Toc23673720)

[**4.1.10** **Oggetto.cpp** 40](#_Toc23673721)

[**4.1.11** **Zaino.h** 40](#_Toc23673722)

[**4.1.12** **Valigia.h** 40](#_Toc23673723)

[**5** **Accorgimenti e risoluzione di bug** 41](#_Toc23673724)

[**6** **Strutture dati intercambiabili** 41](#_Toc23673725)

[**6.1** **Coda con puntatori** 42](#_Toc23673726)

[**6.2** **Coda con priorità tramite lista ordinata** 47](#_Toc23673727)

[**6.2.1** **Cella\_Lista\_Con\_Puntatori\_Monodirezionale.h** 47](#_Toc23673728)

[**6.2.2** **Lista\_Con\_Puntatori\_Monodirezionale.h** 48](#_Toc23673729)

[**6.2.3** **Lista\_Ordinata\_Ereditata.h** 51](#_Toc23673730)

[**6.2.4** **Elemento\_Coda\_Con\_Priorita.h** 52](#_Toc23673731)

[**6.2.5** **Coda\_Con\_Priorita.h** 54](#_Toc23673732)

[**6.2.6** **Servizi\_Coda\_Con\_Priorita.h** 56](#_Toc23673733)

[**7.** **Test** 58](#_Toc23673734)

# **Traccia**

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. Inoltre, è previsto che la realizzazione di almeno due strutture di dati vengano rese intercambiabili.

# **Analisi**

## **Analisi preliminare**

“Adventure with Simunav” è un gioco d’avventura testuale in cui il giocatore controlla il comandante dell’astronave “Neutronia”, il quale dopo essersi svegliato di soprassalto nella sua cabina di pilotaggio, a causa di una elevata temperatura, deve muoversi all’interno dell’astronave risolvendo problemi/enigmi per salvare sé stesso ed il suo equipaggio.

Lo scopo del giocatore è raggiungere la soluzione del gioco risolvendo una serie di enigmi ed evitando vari ostacoli, in quanto il gioco stesso immerge il giocatore in una trama che si genererà passo dopo passo in base alle sue scelte.

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), Sali(A), Scendi(A); di conseguenza 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.

## **Analisi integrativa**

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**

## **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:

## estensionalmente , cioè elencandone tutti i membri,

## 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, e sono unione , intersezione e differenza ben note in matematica. Un insieme che non ha elementi è detto vuoto e viene indicato con Ø.

**Le specifiche: Insieme**

* **Specifica sintattica:**

Tipi: Insieme, Boolean, Tipoelem   
 Operatori [Insieme i]:  
 **creainsieme** : ( ) -> Insieme   
 **insiemevuoto** : (Insieme) -> Boolean  
 **appartiene** : (Tipoelem, Insieme) -> Boolean   
 **inserisci** : (Tipoelem, Insieme) -> Insieme  
 **cancella** : (Tipoelem, Insieme) -> Insieme

**unione** : (Insieme, Insieme) -> Insieme  
 **intersezione** : (Insieme, Insieme) -> Insieme  
 **differenza** : (Insieme, Insieme) -> Insieme

* **Specifica semantica**:   
   Tipi:
  + - *Insieme* : famiglia di insiemi – a definizione estensionale – costituiti da elementi di tipo *Tipoelem.*
    - *Boolean:* insieme dei valori di verità

Operatori [*Insieme i*]:

**creainsieme** = i

Pre: \*\*\* *non ha precondizioni perché può essere sempre generata*

Post:   
 **insiemevuoto** (i) = b   
 Pre: \*\*\* *non ha precondizioni perché può essere sempre generata*  
 Post: b = TRUE se altrimenti b =FALSE  
 **appartiene** (x, i) = i’

Pre: \*\*\* *non ha precondizioni perché può essere sempre generata*

Post: b = TRUE se altrimenti b =FALSE

**inserisci** (x, i) = i’   
 Pre: (OPPURE senza precondizione)

Post:   
 **cancella** (x, i) = i’   
 Pre: (OPPURE senza precondizione)

Post:   
 **unione** (i, i’) = i’’   
 Pre: \*\*\* *non ha precondizioni perché può essere sempre generata*

Post:   
 **intersezione** (i, i’) = i’’   
 Pre: \*\*\* *non ha precondizioni perché può essere sempre generata*

Post:

**differenza** (i, i’) = i’’   
 Pre: \*\*\* *non ha precondizioni perché può essere sempre generata*

Post:

## 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.

## **Le specifiche: Pila**

* **Specifica sintattica:**

Tipi: Pila, Boolean, Tipoelem   
 Operatori [Lista L]:  
 **creapila** : ( ) -> Pila  
 **pilavuota** : (Pila) -> Boolean  
 **leggilpila** : (Pila) -> Tipoelem   
 **fuoripila** : (Pila) -> Pila

**inpila**  : (Tipoelem, Pila) -> Pila

* **Specifica semantica**:   
   Tipi:
  + - *Pila* : l’insieme delle sequenze , di elementi di tipo *Tipoelem,* gestite con accesso LIFO, dove l’elemento i-esimo ha valore
    - *Boolean:* insieme dei valori di verità

Operatori [Pila ]:

**creapila** = P ***EMPTY STACK***  
 Pre:   
 \*\*\* *non ha precondizioni perché può essere sempre generata*

Post:   
 **pilavuota** (P) = b ***EMPTY***  
 Pre: \*\*\* *non ha precondizioni perché può essere sempre generata*  
 Post: b = TRUE se L=   
 **leggilpila** (P) = a ***TOP***  
 Pre:

Post:

**fuoripila** (P) = P’ ***POP***  
 Pre:

Post:   
 **inpila** (a,P) = P’ ***PUSH***  
 Pre:

Post:

# **Implementazione**

## 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

# **Realizzazione**

### **Astro.h**

Sono state aggiunte le seguenti funzioni:



### **Astro.cpp**

Sono stati aggiunti i seguenti vocaboli:

vocabolario.inserisci("valigia", 155);

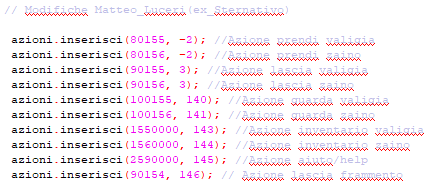
vocabolario.inserisci("zaino", 156);

vocabolario.inserisci("raccogli", 258);

vocabolario.inserisci("aiuto", 259);

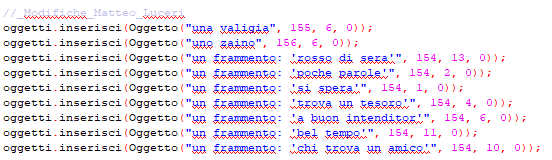
vocabolario.inserisci("help", 259);

Sono stati aggiunte le seguenti azioni:

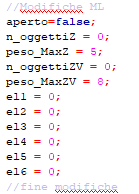


Sono state modificate le seguenti azioni:



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

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.");

}

**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.

# **Accorgimenti e risoluzione di bug**

È stata riscontrata, durante l’analisi, una serie di bug che non permettevano il corretto funzionamento del gioco nel progetto di base (Rosa Chiarappa). Nello specifico:

* Si è riscontrata un evidente disparita fra tutti i codici degli oggetti e delle azioni a loro correlate, fra i progetti. Sono state apportate modifiche ai codici:
  + degli oggetti (salvo quelli standard es *4 tuta*, *11 casco, 23 camince, ecc*);
  + delle azioni, infatti i codici “basati” sul modello LLVVOO non erano espressi nell’ordine delle decine ma delle migliaia. Quindi un ipotetico codice dello Sternativo, *guarda zaino* definita dal codice *10155*, diviene nel progetto base,  *(00)100155*.
* Si è evidenziata un’evidente disparità rappresentativa dei codici degli oggetti nell’applicativo.   
  In particolare, la disparità fra il codice di un vocabolo e il codice *logico* che lo stesso oggetto ricopre nel sistema. Per esempio, il vocabolo *zaino* è salvato nel vocabolario con il codice 155, ma il numero dell’oggetto , nell’elenco degli oggetti dell’applicativo, sarà 144.  
  Per ovviare approssimativamente a ciò si è scelto di commentare il codice enumerando di 5 in 5 tutti gli oggetti nella fase di immissione (Astro.cpp) , per avere un , se pur poco significante, riferimento.

# **Strutture dati intercambiabili**

Si è scelto di implementare la *Coda con puntatori*  nel progetto base perché migliore e completa. Si sono applicati piccole migliorie al fine dell’implementazione.

Per quanto riguarda la seconda struttura dati si è scelto di implementare *una coda con priorità*  realizzata mediante una lista ordinata con puntatori monodirezionale..

Entrambe le realizzazioni sono state aggiunte nella cartella “Strutture intercambiabili” del progetto “base 1911” .

## **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 priorità tramite lista ordinata**

Si fornisce uno schema riassuntivo del processo di creazione della struttura:

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

### **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\_ \*/

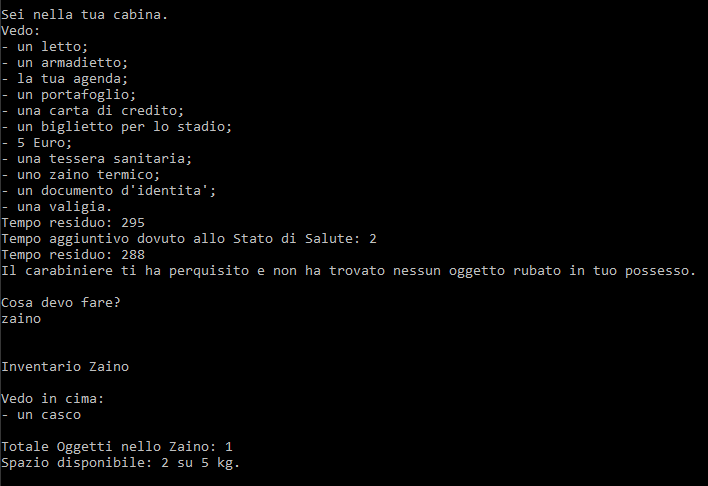
# **Test**

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene sedendo, testo

Descrizione generata automaticamente

Immagine che contiene sedendo

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

Immagine che contiene sedendo

Descrizione generata automaticamente