[21/22] 38090-MOD2 - PROGRAMMAZIONE AVANZATA

Maffeis Isaac 1041473

Relazione di progetto: C++

1. Introduzione

Per il progetto nel linguaggio di programmazione C++ mi sono concentrato nel realizzare un'applicazione che avesse implementato tutti i costrutti studiati a lezione.
In particolare mi sono focalizzato sui concetti di ereditarietà multipla, costruttore e distruttore, campi pubblici e privati, membri virtual e non, overloading, classi astratte, puntatori, oggetto clone, STL (vector, list, iterator), smart pointers.

L'applicazione realizzata (Weekly Planner) si occupa di creare un piano settimanale in base alle esigenze dell'utente, in particolare in primo luogo chiede all'utente informazioni circa la professione, l'età, se svolge attività fisica e quali sono i suoi hobby, successivamente crea e stampa il piano settimanale basandosi su queste informazioni. L'utente a questo punto visiona il piano e può modificare gli eventi (secondari) a proprio piacimento.

È presente una parte di test dove si simula il comportamento dell'utente, facendo varie prove e una parte interattiva da console nella quale si possono testare tutti i vari casi.

2. Caratteristiche

2.1 Funzionalità

L'applicazione è realizzata per organizzare al meglio tutti gli impegni della settimana di una persona, lo scopo principale è quello di fornire al cliente un piano settimanale standard che combaci con il proprio profilo e le proprie passioni, successivamente permettere di modificare il piano con eventi più personali.

Il cliente deve fornire all'applicazione la propria professione, l'età, se svolge attività fisica e i propri hobby, successivamente gli viene presentato un piano settimanale dal lunedì alla domenica con un evento ogni ora. L'utente a questo punto può decidere se tenerlo così, oppure nel caso non sia soddisfatto modificare gli eventi di minore importanza con altri impegni personalizzati.

Il programma deve quindi gestire le interazioni del cliente, creare una lista di giorni per creare la settimana e una lista di eventi per creare un giorno, aggiungere eventi al giorno in base al profilo del cliente e permettere di modificarli successivamente, stampare in modo ordinato queste liste.

3. Composizione

3.1 Strutture dati

Per gestire gli eventi e i giorni della settimana sono state scelte le seguenti strutture di dati proveniente dalla Standard Template Library (STL):

- Vector<T>: array dinamico ad accesso casuale, molto efficace nelle operazioni di inserimento e rimozione in coda, consente di accedere agli elementi in tempo costante.

I contenuti del vettore sono memorizzati in modo contiguo, ogni elemento appena inserito forza lo spostamento a destra dei seguenti elementi, che dipende dalla dimensione del vettore stesso. All'interno del progetto sono presenti due strutture dati di questo tipo, una per memorizzare i puntatori ai giorni della settimana (oggetto) e una per memorizzare gli hobby di una persona (tipo enumerativo), poiché una volta inseriti questi elementi non vengono rimossi o rimpiazzati con altri.

- List<T>: struttura dati in cui gli elementi sono gestiti come una lista doppiamente collegata, tempo costante di inserimento o rimozione in testa, in coda oppure in qualsiasi posto (grazie agli iteratori).
  Tramite questa struttura è stato possibile memorizzare i puntatori agli eventi (oggetto), siccome il programma offre la possibilità di modificarli è quindi richiesto un lavoro di rimozione/inserimento in qualsiasi punto della lista.

3.2 Iteratori

Entità preposta alla scansione di un contenitore secondo le sue modalità di accesso agli elementi, rispettandone vincoli quali la dimensione. Permette scandire la struttura e di effettuare delle elaborazioni a partire dai singoli elementi.
Appartengono alla libreria STL e ce ne sono diversi tipi, nel progetto sono stati usati

- iterator : standard, va dall'inizio alla fine.
- const_iterator : come il precedente, ma non permette modifiche.

3.3 Tipi Enumerativi

I tipi enumerativi permettono di definire tipi che richiedono un numero limitato di valori specifici. In questa applicazione vengono definiti tre tipi enumerativi:

- DayEnum: giorni della settimana
- Priority: Priorità di un evento (primaria, secondaria o terziaria).
- Meal: Pasto (Pranzo o Cena)
- HobbyEnum: Hobby che può scegliere l'utente (+Altro se non è presente).
- Profession: Professione dell'utente (Lavoratore o Studente)

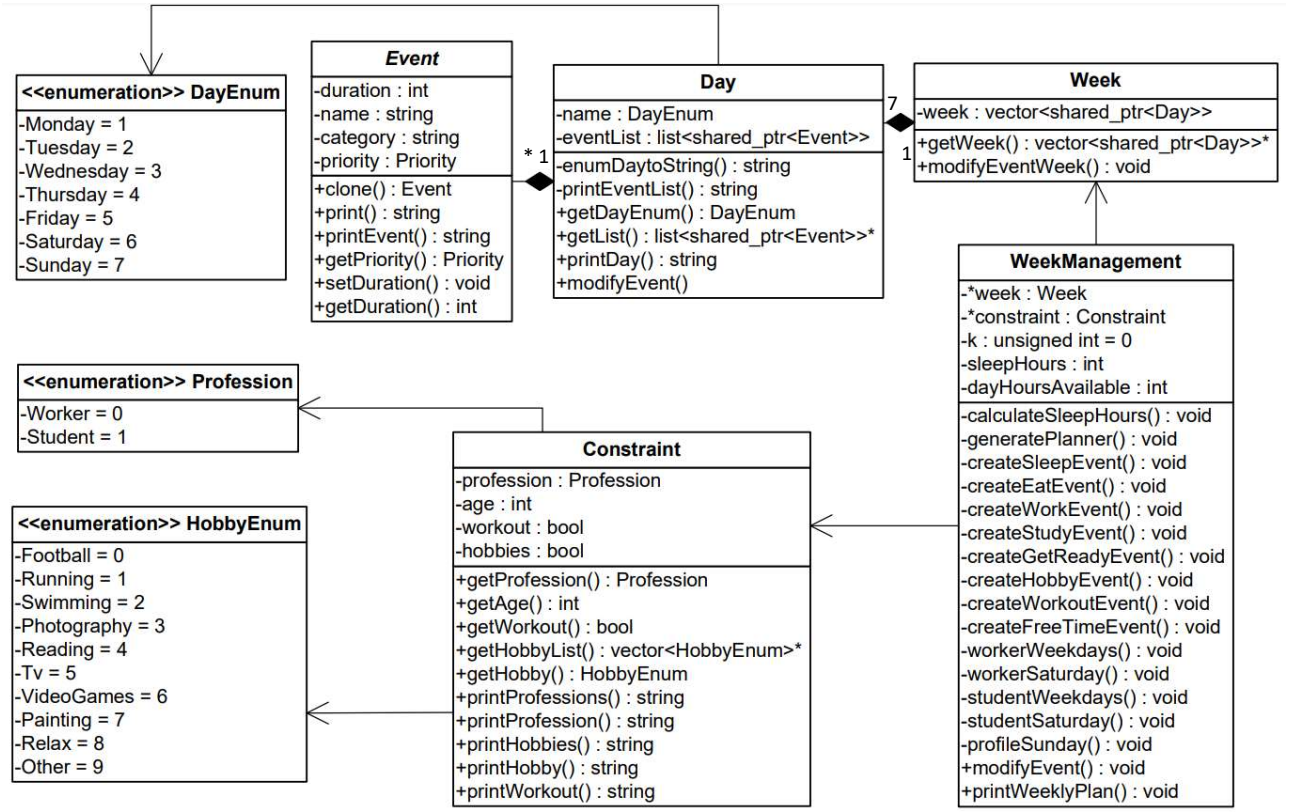
3.4 Copy Constructor vs Funzione Clone

All'interno del programma è stato reso necessario creare un oggetto a partire dal riferimento di un altro oggetto (creare una copia dell'oggetto) e si è dovuto scegliere tra queste due alternative:

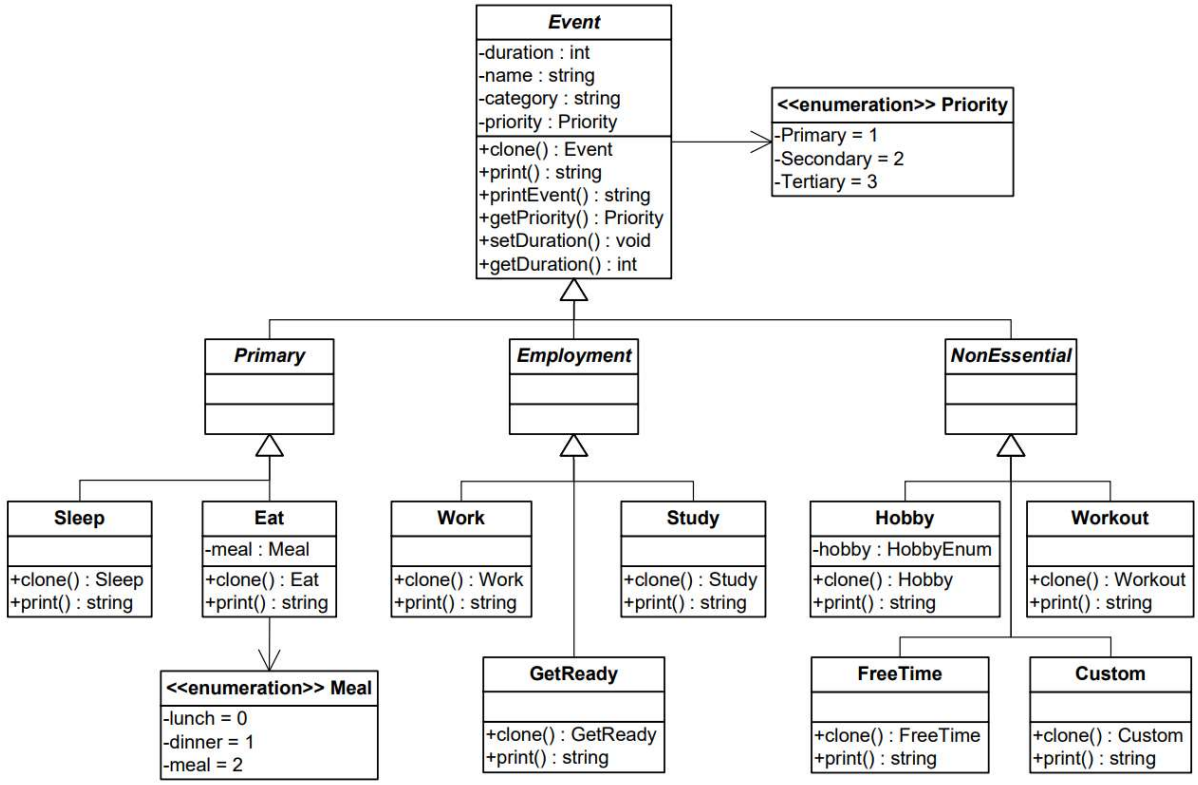
- Copy Constructor : crea un oggetto copia dell'oggetto passato come riferimento nel costruttore, se non è presente l'implementazione nel codice della classe viene usato quello di default.
  Può dare problemi nel caso di polimorfismo ed eredità multipla, perché copia solo i membri che conosce, escludendo quelli delle classi derivate. (Slicing)
- Funzione Clone : Ha la stessa funzione del costruttore copia, ma viene gestita dal programmatore, in particolare viene definita la funzione che restituisce un riferimento all'oggetto della classe nella classe base come virtual, così poi da ridefinirla nelle sottoclassi figlie e restituire l'oggetto opportuno con i rispettivi metodi. Nel progetto è stata scelta la funzione clone poiché si presentava il fenomeno dello slicing con il copy constructor.

3.5 Diagramma delle Classi

Di seguito in figura è rappresentato il diagramma delle classi realizzato in UML:



La classe Event è rappresentata con un alto grado di astrazione, di seguito è mostrata interamente:



3.6 Smart Pointers

Gli smart pointer cercano di risolvere il problema della gestione della memoria (memory allocation e deallocation) per evitare dangling pointers, memory leak etc.

Sono fondamentali per il linguaggio di programmazione RAII (Resource Acquisition Is Initialization).Il principio principale dell'interfaccia RAII è assegnare la proprietà di qualsiasi risorsa allocata all'heap, ad esempio, handle di oggetti di sistema o di memoria allocati dinamicamente a un oggetto allocato nello stack il cui distruttore contiene il codice per eliminare o liberare la risorsa e anche il codice di pulizia associato.

Nel progetto viene usato il tipo di smart pointer shared_ptr :

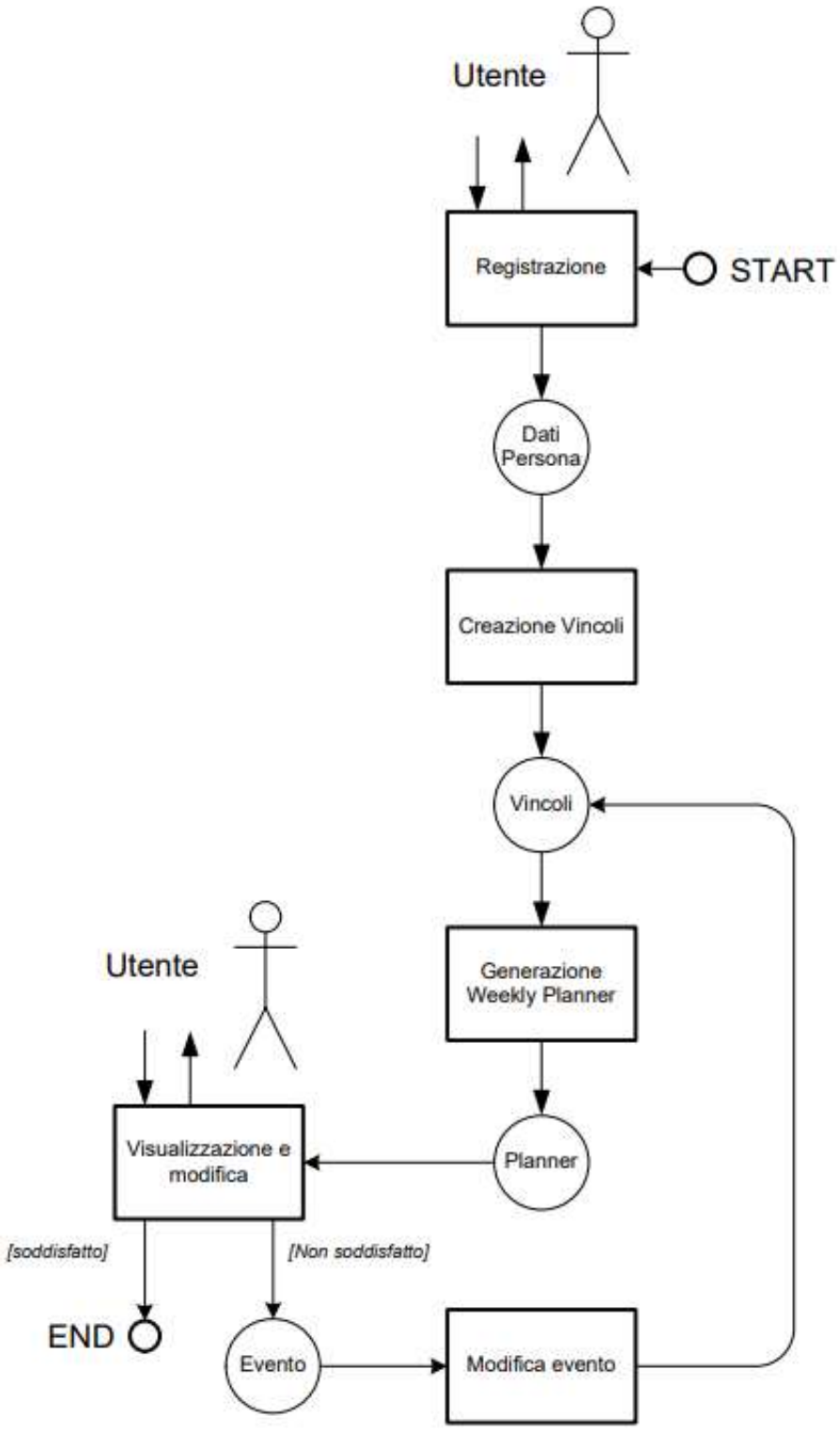
- Puntatore intelligente con conteggio dei riferimenti. Viene utilizzato quando si desidera assegnare un puntatore raw a più proprietari, ad esempio quando si restituisce una copia di un puntatore da un contenitore, ma si desidera conservare l'originale. Il puntatore non elaborato non viene eliminato finché tutti i proprietari di shared_ptr non sono usciti dall'ambito o non hanno ceduto in altro modo la proprietà. Ha le dimensioni di due puntatori.

4. Funzionamento

Il funzionamento è descritto tramite la rete di Petri in figura nella pagina successiva, realizzata con un alto livello di astrazione e assumendo che tutte le transizioni siano realizzate in linguaggio C++, tutte sullo stesso server.

5. Testing

Nella fase di test vengono svolti da prima test funzionali, considerando di creare un planner per ogni profilo disponibile e controllando che il piano settimanale rispetti i vincoli predisposti, successivamente si prende un profilo, si stampa il piano e si vanno a modificare degli eventi, testando i casi più critici (evento a inizio o fine, evento a cavallo di altri eventi, durata diversa), per finire si testano i casi che portano ad eccezioni (modifica di un evento non modificabile, durata superiore). È previsto anche un test interattivo in cui si simula la procedura del cliente tramite input/output da console.


Utente
Registrazione
START
Dati Persona
Creazione Vincoli
Vincoli
Utente
Generazione Weekly Planner
Visualizzazione e modifica
Planner
[soddisfatto]
[Non soddisfatto]
END
Evento
Modifica evento