UNIVERSITÀ DEGLI STUDI DI MILANO
FACOLTÀ DI SCIENZE E TECNOLOGIE

DIPARTIMENTO DI INFORMATICA
GIOVANNI DEGLI ANTONI

Corso di Laurea triennale in
Informatica Musicale

UNA LIBRERIA C PER IL CARICAMENTO E LA MANIPOLAZIONE DI DOCUMENTI IEEE 1599

Relatore: Prof. Federico Simonetta
Correlatore: Prof. Luca Andrea Ludovico

Tesi di Laurea di:
Alessandro Talamona
Matr. Nr. 895744

ANNO ACCADEMICO 2019-2020

Indice

Indice **i**

1 Introduzione **1**
1.1 Scopo Del Software 1
1.2 IEEE 1599 1
1.3 Language Binding 2

2 Tecnologie utilizzate **3**
2.1 C 3
2.2 XML 4
2.2.1 DTD 5
2.3 XPath 5
2.4 Libxml2 5

3 Il Software **6**
3.1 Gestione Del Documento 6
3.1.1 Controllo Degli Input 6
3.1.2 Funzioni Generali 6
3.2 Gestione Dei Livelli 7
3.3 Problemi Riscontrati 8
3.4 Panoramica del progetto 8

Bibliografia **9**

Capitolo 1

Introduzione

Il presente lavoro è organizzato come segue: nel Capitolo 1 ...

1.1 Scopo Del Software

Lo scopo di questo elaborato è quello di costruire una libreria che consenta il caricamento e quindi la manipolazione dei dati contenuti in documenti che rispettino lo standard IEEE 1599, un formato per la rappresentazione musicale sviluppato dal Laboratorio di Infrmatica Musicale (LIM) nel 2008, del quale verranno descritte le caratteristiche di interesse nelle successive sezioni. In particolare si vogliono creare le strutture dati per contenere ogni tipo di informazione ottenibile da tali documenti e le funzioni che permettano il caricamento di tali strutture. Ulteriore obbiettivo è quello di consentire l'utilizzo di questa libreria da diversi linguaggi attraverso le Foreign Function Interface (FFI).

1.2 IEEE 1599

IEEE1599 è un formato basato su XML che descrive contenuti musicali eterogenei in modo completo. In un unico file, simboli musicali, spartiti, tracce audio, performance computer-driven, metadati per la catalogazione, testi e contenuti grafici relativi a una particolare brano musicale vengono collegati e sincronizzati reciprocamente all'interno della stessa struttura. Contenuti eterogenei vengono organizzati in una struttura multi-livello che supporta diversi formati di codifica e un numero di elementi digitali per ogni livello. [1]

I livelli definiti dal formato sono: [2]

- General, che contiene i metadati relativi al brano in oggetto, tra cui le informazioni catalografiche su titolo dell'opera, autori e genere;

- Logic, vero nucleo del formato, destinato alla descrizione simbolica dei contenuti musicali;
- Structural, che identifica gli oggetti musicali su cui il brano è costruito e permette di evidenziarne i mutui rapporti;
- Notational, che contiene le differenti rappresentazioni grafiche della partitura, ad esempio riferibili a diverse edizioni o trascrizioni;
- Performance, che è dedicato ai formati per la generazione di esecuzioni sintetiche da parte dell'elaboratore;
- Audio, che consente di legare al brano in oggetto le esecuzioni audio/video della partitura.

1.3 Language Binding

Con language binding si intende ...

Capitolo 2

Tecnologie utilizzate

In questo capitolo vengono presentate le tecnologie utilizzate per lo sviluppo del software.

2.1 C

Il linguaggio di programmazione scelto è il C in modo da favorire futuri language binding, visto che molti dei linguaggi di alto livello piú utilizzati sono basati su questo linguaggio, come per esempio Python, C++, Java, Javascript e Perl.

Una delle funzionalità piú utilizzate del linguaggio é stata la possibilità di creare tipi di dati strutturati, o strutture, attraverso la parola chiave *struct*. Queste permettono l'aggregazione di più variabili, ma, a differenza degli array, non in modo ordinato e omogeneo poiché una struttura può contenere variabili di tipo diverso.

É possibile definire nuovi tipi di dato anteponendo la parola chiave *typedef* anteposta alla dichiarazione di una struttura e specificando il nome con cui riferircisi. Agli elementi del tipo di dato così creato si puó accedere tramite l'oeratore ‘‘ . ’’

typedef struct elemento int informazione; nome_tipo;

Tra i concetti fondamentali del linguaggio C c'è quello di *puntatore*, perché permette, tra le altre cose, di poter astrarre in modo semplice strutture dati complesse. Un puntatore è una variabile che contiene l'indirizzo di memoria di un'altra variabile e si dichiara anteponendo il carattere * al nome della variabile. L'asterisco é anche l'operatore di indirezione, cioè restituisce l'oggetto puntato dal puntatore. L'operatore *&*, davanti a un nome di variabile, restituisce invece l'indirizzo di quest'ultima.

I puntatori sono strumenti necessari anche per la gestione delle *liste*, collezioni di un numero di elementi omogenei di cui non si conosce a priori la numerosità che può anche variare nel tempo. Inoltre, al contrario di quanto accade per gli array, tali elementi non occupano posizioni contigue in memoria. Una lista, in C, puó essere vista come una struttura che contiene campi di informazioni e almeno un puntatore a cui viene legato l'elemento successivo.

struct elemento int informazione; struct elemento *puntatore; [1]

Il C permette di gestire l'allocazione della memoria in modo dinamico, assegnando alle variabili solamente la quantitá di memoria necessaria. Per fare ció, esistono funzioni come *malloc* e *calloc*, adibite all'allocazione della memoria , *realloc*, per consentire la modifica di spazi di memoria precedemtemente allocati, e *free* per liberare la memoria allocata.

Nel capitolo relativo allo sviluppo del software verranno discussi alcuni esempi in cui verranno utilizzate tutte queste funzionalitá.

2.2 XML

Extensible Markup Language (XML) è un linguaggio di markup per documenti che contengono informazioni strutturate, quindi definite non solo dal contenuto, ma anche dal ruolo occupato da tale contenuto. I linguaggi di markup sono insiemi di regole che definiscono dei meccanismi attraverso i quali é possibile identificare una struttura in un documento. [3] Applicato al nostro caso, ciò vale a dire che la struttura di un documento IEEE1599 come presentata nell'introduzione di questo lavoro corrisponderebe a:

```
<ieee1599>
        <general>...<\general>
        <logic>...<\logic>
        <structural>...<\structural>
        <notational>...<\notational>
        <performance>...<\performance>
        <audio>...<\\audio>
</ieee1599>
```

[1] https://www.html.it/pag/15418/introduzione-alle-liste/

2.2.1 DTD

Un Document Type Definition (DTD) definisce la struttura di un documento XML, specificando quali elementi e attributi sono ammessi e come devono essere organizzati. Un documento XML, per essere *valido*, deve rispettare le regole del DTD relativo oltre a dover essere ben formato, ovvero rispettante le regole di sintassi di XML. Il DTD che definisce la struttura dei documenti IEEE 1599, sul quale si basano le scelte di programmazione effettuate per questo elaborato, è disponibile sul sito ufficiale del Computer Society of the Institute of Electrical and Electronics Engineers [2], attualmente alla versione del 2008. Il DTD é dichiarato all'interno del documento XML nel tag vuoto *DOCTYPE*. Solitamente è definito in un file esterno e nei documenti viene espresso come riferimento a una risorsa esterna, come mostrato nell'esempio seguente.

```
<!DOCTYPE ieee1599 SYSTEM "http://standards.ieee.org/downloads/1599/1599-
```

2.3 XPath

2.4 Libxml2

Il software creato si appoggia sulla libreria libxml2 [3] per il parsing dei file XML generici. Sul sito è presente un tutorial [4] su come utilizzare i metodi principali del linguaggio dal quale sono state estrapolate alcuni algoritmi ricorrenti per le operazioni di parsing. Questa libreria, tra le varie disponibili, è una tra le piú utilizzate perché puó facilitare la programmazione basandosi sulla creazione in memoria di un albero a partire dal document XML, Supporta XPath e permette la validazione tramite DTD durante il parsing.

[2] https://standards-ieee-org.pros.lib.unimi.it/downloads.html
[3] http://www.xmlsoft.org/
[4] http://xmlsoft.org/tutorial/

Capitolo 3

Il Software

In questo capitolo viene descritto il metodo con cui è stato sviluppato il software, ...

3.1 Gestione Del Documento

3.1.1 Controllo Degli Input

Per prima cosa bisogna analizzare le caratteristiche del tipo di documento su cui bisogna lavorare. Il software deve ricevere in input un file XML e ci sarà quindi bisogno di controllare che il file sia ben formato, ma visto che si ha a disposizione un DTD, sarebbe ancora meglio lavorare su un file valido.

La libreria libxml2 mette a disposizione il metodo xmlValidateDTD per verificare che un documento XML sia valido e su questa funzione è stata sviluppata una parte di codice, contenuta in common.c, che controlla la validitá del documento IEEE 1599.

Un altro controllo richiesto da libxml2 per poter operare sui documenti XML è quello di verificare che i caratteri usati all'interno del documento si basino sulla codifica UTF-8. Anche per questo controllo é stata sviluppata una parte di codice in common.c, basandosi sull'algoritmo d'esempio disponibile direttamene sul tutorial ufficiale di libxml2 relativo a questo argomento.

3.1.2 Funzioni Generali

Le due funzioni di base, sviluppate sull'utilizzo di libxml2, sono quelle per il caricamento del documento in memoria e per la navigazione di tale documento tramite XPath e sono contenute nel file common.c.

La funzione getDoc, ricevendo come input il percorso al documento ieee1599 da elaborare, utilizza la funzione xmlParseFile per creare in memoria una struttura ad

albero che rispecchia quella del documento XML e fornisce l'indirizzo del puntatore alla radice di tale nodo. Questa funzione viene utilizzate dal main del programma ogni qual volta l'utente sceglie un file da caricare.

La funzione getNodeset, avendo giá un documento caricato in memoria, permette di estrapolare un contesto a partirte da un'espressione XPath.

Entrambe queste funzioni utilizzano algoritmi e funzioni di libreria ampiamente documentati sul sito libxml2, nello specifico sono stati utili gli esempi del tutorial di base.

Altre funzioni utili contenute in common.c sono concat, che permette di concatenare due stringhe, xmlCharToInt e xmlCharToDouble. Le ultime due vegono utilizzate ogniqualvolta ci sia il bisogno di caricare valori numerici contenuti nel documento XML da caricare. In caso di sviluppi futuri, qui è dove sarebbe meglio introdurre nuovi funzioni di conversione per altri tipi, anche complessi.

Il modulo fileChooser.c contiene tutte le funzioni che permettono all'utente di scegliere il documento da caricare. All'utente vengono mostrati tutti i file XML presenti nella cartella File presente nella cartella del programma e viene richiesto di scegliere il file su cui lavorare. Al termine dell'elaborazione si può scegliere se terminare l'esecuzione o continuare con un altro documento.

3.2 Gestione Dei Livelli

Si è deciso di tradurre ogni elemento presente nella gerarchia XML in una struct il cui contenuto sono i suoi attributi. [esempio] Non tutte le Entity citate nel DTD sono state trasformate in struct: spesso è stato possibile sostituire un elemento con una lista di suoi figli. [esempio] Le struct impostate come liste sono state utilizzate anche per tutti quegli elementi che possono comparire più volte come figli di un dato elemento. [esempio]

I tipi di dato utilizzati all'interno delle struct sono per la maggior parte stringhe, dato che la natura delle informazioni contenute nei documenti XML non è definibile a priori. In alcuni casi, invece, altri tipi di dato più adatti, per esempio sono stati utilizzati tipi numerici dove i valori non potevano che essere valori su cui era possibile fare operazioni: un esempio sono num e den che rappresentano le frazioni indicanti la durata delle note, oppure il timing degli event espresso in VTU. [esempio]

Analizzando i valori ammessi dal DTD per alcuni attributi,è stato possibile definire alcune enumerazioni utilizzabili per sviluppi futuri del software: per esempio possono rivelarsi utili per i controlli relativi all'input di certi valori, limitando la scelta ai soli valori enumerati, se si decide di sviluppare funzioni che permettono di modificare le strutture caricate.

Lo scheletro del software è stato modellato suddividendo il codice in diversi moduli per rispecchiare la struttura ad albero definita dal DTD. Partendo dalla radice del documento arrivando fino all'ultimo elemento, è stato possibile costruire una gerarchia di strutture ad albero che rispecchia la struttura definita dal DTD. La struct radice, chiamata ieee1599 contiene le strutture general, logical, structural, notational, performance e audio, come mostrato dal seguente schema. [schema] Data la mole di informazioni possibili contenuto nel layer logical e in particolare nella componente LOS,è stato deciso di suddividere ulteriormente il codice per contenere gli elementi di quest'ultima (managerLosElements).

Moduli dei layer Definiti gli header per i moduli dei livelli,è stato possibile definire le funzioni per il caricamento delle strutture. Ogni layer ha la sua funzione loadNomeLayer che, utilizzando xPath, recupera il nodeset contenente tutti gli elementi appartenenti a tale layer. Il nodeset viene scandito elemento per elemento e ogni volta che un elemento potrebbe contenerne altri viene fatta la scansione dei suo figli, fino a quando si arriva a una foglia, dopodichè si torna indietro. [cercare nome dell'algoritmo] [codice] La struttura del codice di cui sopra è comune a quella utilizzata per eseguire tale operazione per tutti gli elementi del documento, adattata a seconda della presenza di più o meno attributi, più o meno elementi figli e livelli di profondità da analizzare.

3.3 Problemi Riscontrati

Il formato IEEE1599 utilizza i DTD esterni per SVG e MIDI.

Problema SVG Nel LOS del livello logical, gli elementi custom_articulation, custom_hsymbol e slur contengono SVG.... Problema MIDI Nel livello perofmrance, gli elementi sys_ex e midi_event contengono sysex e midi channel message ... Inoltre nel livello structural, gli elementi feature_object contengono added_feature_object_class.....

3.4 Panoramica del progetto

Prima di addentrarsi nei dettagli è bene fornire una panoramica (anche molto schematica, corredata da un diagramma) del lavoro svolto, in modo che il lettore abbia una mappa concettuale con cui orientarsi.

Bibliografia

[1] Luca A Ludovico. Key concepts of the ieee 1599 standard. In *Proceedings of the IEEE CS Conference The Use of Symbols To Represent Music And Multimedia Objects, IEEE CS, Lugano, Switzerland*, pages 15–26, 2008.

[2] Adriano Baratè, Luca A Ludovico, and Davide A Mauro. Tecnologie basate su xml per la fruizione avanzata dei contenuti musicali. In *Conferenza GARR_09 Selected papers*, page 43, 2010.

[3] Norman Walsh. What is xml. *XML. commune*, 1998.

Progetto sviluppato presso il Laboratorio di Informatica Musicale
`https://www.lim.di.unimi.it`