Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ОПИС МУЗИКИ В HASKELL

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
кандидат фізико-математичних наук,
доцент Проценко В. С.

_____

(підпис)

“___._.” _____ 2020 р.

Виконала студентка Чумаченко О. Р.

“___._.”_____ 2020 р.

Київ 2020

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри інформатики,

к. ф.-м. н. С. С. Гороховський

_____

(підпис)

“____”_____2020 р

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студентки Чумаченко Олександри Романівни факультету інформатики 3 курсу

Тема: Опис музики в Haskell

Зміст ТЧ до курсової роботи:

Анотація

Вступ

Розділ 1. Аналіз предметної області

Розділ 2. Розробка

Розділ 3. Результати

Розділ 4. Проблеми, що виникали під час роботи

Висновки

Список літератури

Дата видачі “___” _____ 2020 р. Керівник _____

(підпис)

Завдання отримав _____

(підпис)

Тема: Опис музики в Haskell

Календарний план виконання курсової роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	01.11.2019	
2.	Пошук тематичної літератури за темою роботи	03.12.2019	
3.	Ознайомлення з технічною літературою за темою роботи	20.01.2020	
4.	Ознайомлення з основами нотного запису	06.02.2020	
5.	Виконання аналізу сучасних інструментів для опису музики	20.02.2020	
6.	Повторення синтаксису Haskell	03.03.2020	
7.	Аналіз існуючих бібліотек для опису музику на мові Haskell	15.03.2020	
8.	Вибір музичних творів для опису	18.03.2020	
9.	Написання практичної частини курсової роботи	06.04.2020	
10.	Написання текстової частини курсової роботи	30.04.2020	
11.	Написання висновків	03.05.2020	
12.	Створення презентації	09.05.2020	
13.	Захист курсової роботи	20.05.2020	

Зміст

Анотація

В сучасному світі цифрових технологій опис музики програмними засобами – тема, що цікавить багатьох композиторів та науковців. Дана робота присвячена опису музики в мові Haskell.

Під час роботи були проаналізовані існуючі інструменти для опису музики та розглянуті деякі функції та типи даних бібліотеки Euterpea.

Результатом роботи є створені програми, що можуть відтворити описаний в них музичний твір.

Вступ

Опис музики за допомогою програмних засобів – цікава та обширна тема, яка потребує знань як в програмуванні, так і в основах музичної теорії.

Питання, пов'язані з програмуванням музики, обговорюються досить довгий час. З символікою певних чисел пов'язували форму своїх творів нідерландські майстри строгого стилю. Чайковський вважав, що будь-яка гарна музика - програмна. Шоста симфонія Бетховена і симфонія "Манфред" Чайковського - класичні приклади програмної музики: їх зміст і форма були запрограмовані композиторами на основі зовнішніх образних сюжетів.[1] Сьогодні існують майстер-класи по написанні електронної музики використовуючи мову програмування Haskell [2]

Мета курсової роботи полягає у ознайомленні з можливостями, що існують в мові Haskell для опису музики та отриманні практичного досвіду у роботі з ними.

Завданням курсової роботи є створення програми, що відтворює музичний твір.

Об'єктом дослідження є можливості мови програмування Haskell для опису та створення музичних творів.

Предметом дослідження є практичне використання бібліотеки Euterpea для опису музики на Haskell.

Практичним значення одержаних результатів є особистий досвід та опис можливостей, що надає бібліотека Euterpea для опису музичних творів.

Практична частина написана на мові Haskell з використанням бібліотеки Euterpea. Для запуску використовується Glasgow Haskell Compiler's interactive environment(GHCi).

Робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел.

Перший розділ присвячено аналізу предметної області. Розглянуто сучасний стан питання та виконано огляд існуючих інструментів для опису музики.

У другому розділі наведено основні теоретичні відомості необхідні для опису музики, використовуючи мову програмування Haskell.

Третій розділ присвячено результатам курсової роботи.

У четвертому розділі описані основні проблеми, що виникали під час роботи.

Створено програмні продукт, які програють запрограмовану в них музику.

Постановка задачі

Реалізація програмного продукту на мові Haskell, який за допомогою GHCi програє описану в ньому мелодію.

Розділ 1. Аналіз предметної області

1.1 Аналіз сучасного стану питання

Питання програмного опису та створення музичних творів стає дедалі більш популярним. Композиторів довгий час приваблює ідея інструменту для автоматичного створення музики. [3] В роботах відомих композиторів, таких як Йоганн Себастьян Бах та Вольфганг Амадей Моцарт, навіть спостерігали використання алгоритмічних методів. [4]

На сьогодні існує безліч наукових робіт, які тим або іншим чином висвітлюють питання опису та створення музики, пропонують нові фреймворки, бібліотеки, надбудови [6,8] та пояснюють основні принципи роботи з ними. [5, 7]

1.2 Огляд існуючих інструментів розробки

Процес створення музики все більше потребує програмної підтримки. На сьогоднішній день існує багато можливостей та інструментів для програмного опису музичного твору. Багато програм пропонують функції для музичної нотації та відтворення, які замінюють традиційні. Деякі з них можуть навіть автоматично створювати частини композиція, такі як ритмічний супровід. Створення музики популярне застосування Haskell, існує декілька бібліотек для опису та генерації музичних творів. [9]

1.2.1 Haskore

Haskore - це бібліотека Haskell, призначена для вираження музичних структур у високорівневому, декларативному стилі функціонального програмування.

У Haskore музичні об'єкти складаються з примітивних понять, таких як ноти і паузи, операції перетворення об’єктів, такі як масштабування темпів, та операції поєднання музичних об’єктів, для утворення складніших, такі як паралельні та послідовні поєднання. [11]

Hascore організований наступним чином:

1. Структура даних для абстрактного опису музики;
2. Виконавча функція, що перетворює цю структуру даних у послідовність музичних подій;
3. Конвертація такої послідовності у потік MIDI, аудіо потік, або CSound файл. [13]

1.2.2 Euterpea

Euterpea - кросс-платформна, предметно-орієнтовна мова для програм комп'ютерної музики, вбудована в мову програмування Haskell. Підходить для представлення музики високого рівня, алгоритмічної композиції, аналізу музики, роботи з MIDI, низькорівневої обробки звуку, синтезу звуку та дизайну віртуальних інструментів.

Була створена Полом Худаком і розроблена групою Yale Haskell. Є нащадком бібліотек Haskore та HasSound.

На даний час досягла стабільної фази розвитку, тобто не зазнає значних змін коду. Основні функції бібліотеки лишаються незмінними. [10]

Euterpea поділена на дві основні частини: рівень нот та рівень сигналів.

На рівні нот можна створювати музику, яка може бути різною за рівнем складності: від простих творів до симфоній.

На сигнальному рівні можна створювати нові інструменти та оброблювати вже існуючий музичний запис.

1.2.3 Kulitta

Kulitta - це фреймворк для автоматизації і алгоритмізації музичних композицій, що базується на мові програмування Haskell. Алгоритми для музичних композицій використовуються для автоматичного створення музики, або як інструмент, який робить можливим для людини створювати насичені та комплексні роботи. Алгоритми для генерування музики виробляють широкий діапазон результатів від високоматематичних та сучасних до дуже традиційних.

Компоненти Kulitta поділяються на три основні категорії:

1. Абстрактна/структурна генерація: набір моделей та алгоритмів для ітеративної генерації гармонійних структур або інших абстракцій.
2. Музична інтерпретація: математичні моделі та алгоритми задоволення обмежень абстрактних музичних функцій.
3. Навчання: алгоритми навчання в автономному режимі, щоб отримати виробничі ймовірності для музичних граматик, які потім можуть бути використані для створення абстрактної структури. [5]

1.3 Висновок за главою 1

Для розробки практичної частини була обрана бібліотека Euterpea, оскільки вона відповідає вимогам поставленого завдання і містить усі необхідні компоненти для його реалізації.

Розділ 2. Розробка

2.1 Необхідне програмне забезпечення

Для реалізації практичної частини роботи використовується Haskell Platform, бібліотека Euterpea та будь-який текстовий редактор, наприклад, Notepad.

Відповідно до рекомендацій з офіційного сайту[10], бібліотека встановлюється через командний рядок з використанням наступних команд:

```
cabal update
cabal install Euterpea
```

Для перевірки коректності встановлення бібліотека необхідно виконати в GHCi наступні команди:

```
import Euterpea
play $ c 4 qn
```

В результаті виконання має прозвучати один звук.

2.2 Типи та функції, що використовувались

Для опису музичних творів під час реалізації практичної частини курсової роботи найчастіше мною використовувався їх нотний запис. Тому необхідним було розуміти основні його складові, та яким чином можна передати їх використовуючи Euterpea.

Нотний запис - європейська система нотації, що існує з XVII століття. До її складу входять такі основні елементи: нотний стан(нотоносець) з п'яти паралельних горизонтальних ліній, на якому розміщують різноманітні знаки нотопису.[14]

2.2.1 Ноти

Нота (італ. nota - позначка) - знак, який застосовується для запису музичних звуків, позначає їх висотне положення на нотоносці та тривалість, має вигляд незатушованого, або затушованого овалу. Тривалості, менші за четвертну, позначаються хвостиками (прапорцями) або в'язками. [14]

В Euterpea можна створити ноту використовуючи тип даних Primitive:

data Primitive a = Note Dur a

Але цей тип даних не дозволяє відтворювати (програвати) ноти. Для цього визначений тип data Music a = Prim (Primitive a)

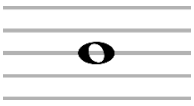
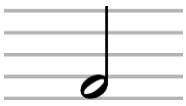
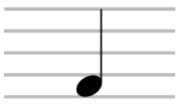
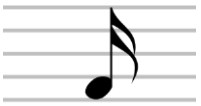
2.2.1.1 Тривалість

Тривалість – час звучання. [14]

В бібліотеці Euterpea тривалість представлена наступним типом:

type Dur = Rational

Для типу Dur визначено наступні скорочення: wn, hn, qn, en, sn, dwn, dhn, dqn, den, dsn, ddqn, ddhn, dden. Окрім скорочень є можливість записувати як звичайне раціональне число: 3, 2.5, 1/4.

				
wn	hn	qn	en	sn
1/1	1/2	1/4	1/8	1/16

Таблиця 2.1 Відповідність між деякою тривалістю, визначеними скороченнями та позначенням ноти

2.2.1.2 Висота

Висота звуку - одна з основних акустичних якостей звуку. Атрибут, який дозволяє впорядковувати звуки за частотною шкалою від низьких до високих. [15] Висота звуку є наслідком сприйняття людиною частоти коливань вібратора та безпосередньо залежить від неї. [14]

В бібліотеці Euterpea класи висоти звуку представлені наступним типом:

PitchClass – це символьне представлення класів висоти звуку.

data PitchClass = Cff | Cf | C | Dff | Cs | Df | Css | D | Eff | Ds | Ef | Fff | Dss | E | Ff | Es | F | Gff | Ess | Fs | Gf | Fss | G | Aff | Gs | Af | Gss | A | Bff | As | Bf | Ass | B | Bs | Bss

s – позначає дієз, ss – подвійний дієз, f – бемоль, ff – подвійний бемоль.

Висота звуку представлена типом Pitch – кортежем, що складається з класу висоти та октави :

type Pitch = (PitchClass, Octave).

Октава – це частина звукоряду, яка містить сім діатонічних або дванадцять хроматичних ступенів. [14]

В Euterpea тип Octave є синонімом до типу цілих чисел, та може набувати значення від мінус 1 до 9.

2.2.1.3 Гучність

В багатьох творах для опису їх нот буде достатньо вказати клас висоти, октаву та тривалість. Але в інших необхідним атрибутом є гучність.

Гучність - слухове уявлення про силу звука, що виникає в свідомості людини під час сприйняття звука і залежить від амплітуди коливань вібратора, відстані від джерела звука, частоти коливань. [14]

Для опису гучності в бібліотеці запропонований тип type Volume = Int, який приймає значення від 1 до 127.

Приклад використання типу Volume під час створення ноти:

v_note:: Music ((PitchClass, Octave), Volume)

v_note = Prim (Note qn ((C,4 :: Octave), 100::Volume)) або
v_note = note qn ((C,4 :: Octave), 100::Volume)

2.2.1.4 Приклад опису ноти

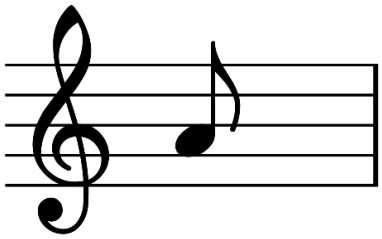


Рис. 2.1 Нота ля першої октави

На початку рис. 2.1 зображено скрипковий ключ. Він розташовує ноту соль першої октави на другій знизу лінійці нотного стану. За ним йде нота ля тривалістю 1/8. В Euterpea дану ноту можна записати наступним чином:

note_la = Prim (Note en (A, 4))
або можна скористатись більш коротшим способом запису:
note_la = a 4 en.

2.2.2 Паузи

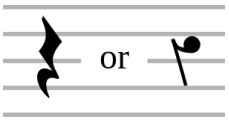
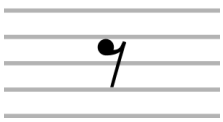
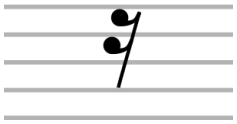
Пауза (від грец. pausis - перерва, зупинка) - 1) тимчасова перерва в звучанні одного, кількох або всіх голосів музичного твору, що триває

протягом певного часу. 2) Знак, що визначає тривалість паузи в звучанні. За тривалістю паузи є такими ж, як і ноти. [14]

В Euterpea можна створити паузу використовуючи тип даних Primitive:

data Primitive a = Rest Dur

А для відтворення пауз необхідно скористатись типом data Music a = Prim (Primitive a).

				
wn	hn	qn	en	sn
1/1	1/2	1/4	1/8	1/16

Таблиця 2.2 Відповідність між деякою тривалістю, визначеними скороченнями та позначенням паузи

Створити паузу тривалістю 1/2 можна наступним чином:

pause = Prim (Rest qn) або pause = Prim (Rest (1/2))

аналогічно до нот існує більш короткий спосіб створення паузи:

pause = rest qn або pause = rest (1/2).

2.2.3 Поєднання кількох нот або пауз

Музичний твір — це твір, у якому образи виражені за допомогою звуків, які в свою чергу представленні у вигляді нот. Ноти можуть відтворюватись по черзі або одночасно, тому вміти поєднувати ноти та паузи необхідно для опису музики.

Для таких цілей бібліотека Euterpea надає оператори для послідовного і паралельного поєднування нот.

2.2.3.1 Послідовне поєднання

Для послідовного поєднання кількох нот та пауз існує спеціальний оператор:

(:+:) :: Music a -> Music a -> Music a

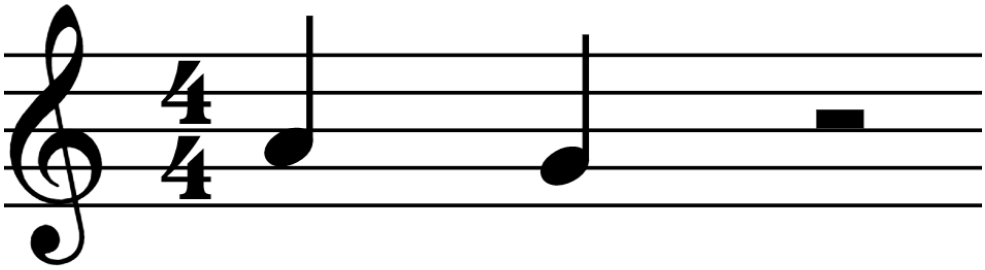


Рис. 2.2 Приклад послідовного поєднання

Приклад зображений на рис. 2.2 можна реалізувати як

exp1 = a 4 qn :+: g 4 qn :+: rest hn

Також для послідовного поєднання кількох нот або пауз можна використовувати функцію line :: [Music a] -> Music a. Тому приклад, зображений на рис.2 можна реалізувати наступним чином:

exp1_1 = line [a 4 qn, g 4 qn, rest hn]

2.2.3.2 Паралельне поєднання

Для паралельного поєднання використовується оператор:

(:=:) :: Music a -> Music a -> Music a

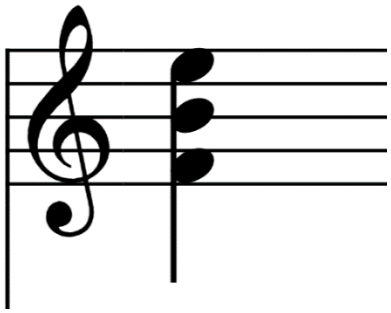


Рис. 2.3 Приклад паралельного поєднання

Приклад зображений на рис. 2.3 за допомогою бібліотеки Euterpea реалізовується наступним чином:

exp2= f 4 qn :=: b 4 qn :=: e 5 qn

Аналогічним чином можна використовувати і функцію chord :: [Music a] -> Music a :

exp2_2 = chord [f 4 qn, b 4 qn, e 5 qn]

2.2.4 Музичний розмір

Розмір – кількісна характеристика тактового метру, що показує число ритмічних долей у такті і записується як дріб без риски між чисельником і знаменником на початку нотного тексту або під час зміни метра. [14]

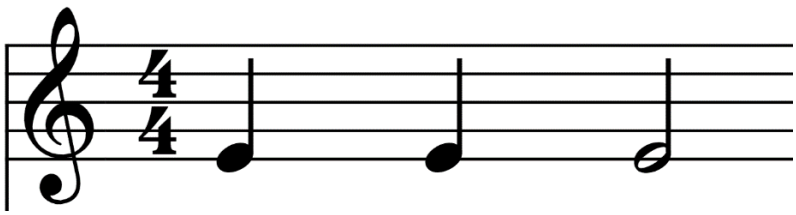


Рис. 2.4 Приклад позначення розміру

Для реалізація рис. 2.4 в Euterpea запропонована функція tempo :: Dur -> Music a -> Music a. Опис частини музичного твору, що запропонований на рис. 4 матиме вигляд:

ex3 = tempo(4/4) (e 4 qn :+: e 4 qn :+:e 4 hn).

Також часто замість двох цифр використовуються наступні позначення:

Рис. 2.5 Позначення що використовується замість 4/4

Рис. 2.6 Alla breve, позначення що використовується замість 2/2

2.2.4.1 Помітка метронома

Також існує символ, написаний на початку партитури, який при будь-якій значній зміні темпу, точно визначає темп музики, присвоюючи абсолютну тривалість усім нотним значенням у партитурі.

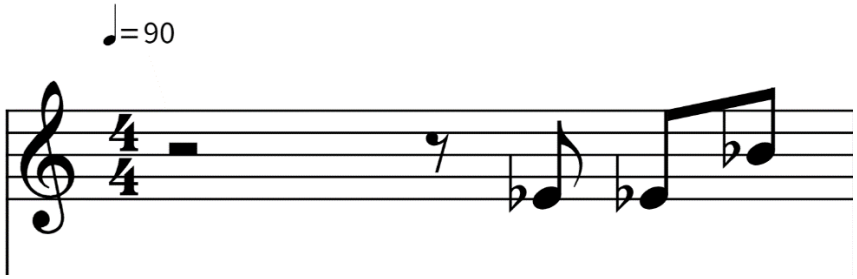


Рис. 2.7 Приклад позначки темпу

За замовчуванням функція відтворення play в Euterpea використовує темп, еквівалентний 1/4 ноти, має значення метронома 120. Тому темп слід масштабувати відповідно до коефіцієнта. [16]

Для прикладу з рис. 2.7 коефіцієнт буде: (qn /qn) ∗ (90/120). Відповідно реалізація цього прикладу матиме вигляд:

ex4 = tempo ((qn /qn) ∗ (90/120)) (tempo (4/4) (rest hn:+: rest en :+: ef 4 en :+: ef 4 en :+: bf 4 en)).

2.2.5 Ключові знаки

Ключові знаки - знаки альтерації, виставлені біля ключа, що вказують тональність твору або його частин. [14]

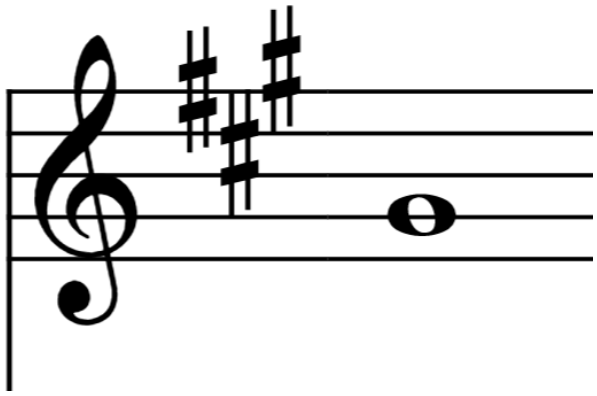


Рис. 2.8 Ля-мажор

Для реалізації в бібліотеці можна використати функцію keysig :: PitchClass -> Mode -> Music a -> Music a. Прикладом її використання є :

exp5 = keysig A Major (g 4 wn)

2.2.6 Динаміка

Динаміка - одна зі сторін організації музики, тісно пов'язана з силою звучання, дією різних акустичних компонентів. [14] Бібліотека Euterpea задовольняє реалізацію динаміки за допомогою типів даних:

data Dynamic = Accent Rational| Crescendo Rational| Diminuendo Rational | StdLoudness StdLoudness | Loudness Rational

data StdLoudness = PPP | PP | P | MP | SF | MF | NF | FF | FFF

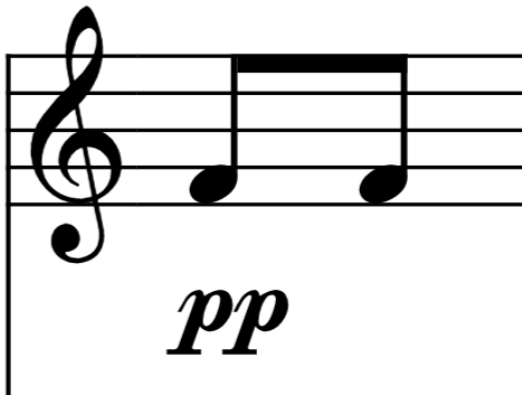


Рис. 2.9 Піанісимо (дуже тихо)

Реалізувати приклад, що зображений на рис. , можна за допомогою функції phrase :: [PhraseAttribute] -> Music a -> Music a:

```
ex6 = phrase [Dyn (StdLoudness PP)] (e 4 en :+: e 4 en)
```

2.2.7 Повтори

Досить часто в музичних творах виникає необхідність в повторенні частини твору, або деяка комбінація нот декілька разів повторюється одна за одною. В таких випадках, для уникнення копіювання одного і того самого коду можна використовувати надану бібліотекою Euterpea функцію times :: Int -> Music a -> Music a, що повторює необхідний фрагмент задану кількість раз.

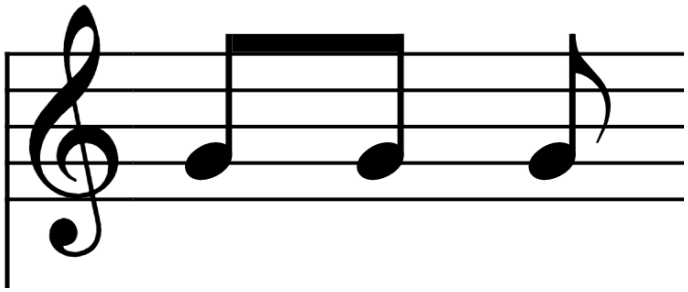


Рис. 2.10 Повторення однієї ноти

На рис. 2.10 наведено приклад який можна реалізувати наступним чином:

```
ex7 = g 4 en :+: g 4 en :+: g 4 en
```

або з використанням функції times:

```
ex7 = times 3 (g 4 en).
```

2.2.8 Функція відтворення

Для відтворення описаної мелодії бібліотека Euterpea пропонує функцію play :: (ToMusic1 a, Control.DeepSeq.NFData a) => Music a -> IO (). Використовується вона наступним чином:

```
> play ex5.
```

2.2.9 Вибір інструменту

Іноді виникає необхідність в програванні різних частин твору різними інструментами, або твір найкраще звучить лише при використанні певного інструменту. Euterpea пропонує на вибір більше ста різних інструментів та надає можливість використовувати власні. Змінити інструмент можна за допомогою функції instrument :: InstrumentName -> Music a -> Music a:

```
ex8 = instrument BrightAcousticPiano ex5
```

2.3 Висновок за главою 2

Нотний рівень бібліотеки Euterpea повністю задовольняє потреби для опису та відтворення нескладних музичних творів. Окрім розглянутих у роботі можливостей, існують типи даних та функції, що дозволяють описувати складнішого рівня твори, з використанням різного виду орнаментики.

Розділ 3. Результати

Під час виконання роботи були проаналізовані існуючі приклади опису музики наведені в книзі “Haskell School of Music”, [16] а також в ресурсах мережі інтернет. [12, 17]

Спираючись на ці джерела були створені програми, які містять опис наступних музичних творів: “Jingle Bells”, “Imagine Dragons – Demons”, “Танець маленьких каченят”. Для реалізації цих програм було достатньо використовувати функції лише з нотного рівня бібліотеки Euterpea.

Розділ 4. Проблеми, що виникали під час роботи

Основними труднощами під час роботи над реалізацією практичної частини були відсутність освіти в музичній сфері і, як наслідок, складнощі під час ознайомлення з роботою функцій бібліотеки та під час читання нотного запису для опису музики.

Вирішення даної проблеми полягало у більш детальному ознайомленні з основами музичної теорії та нотного запису. В деяких випадках вирішенню проблеми сприяла консультація з людьми, які мають освіту та досвід в цій сфері.

Окремо треба виділити також, що під час аналізу існуючих способів опису музики на Haskell, часто можна було знайти приклади використання бібліотеки Euterpea, але більш старої версії, в якій ще існували необхідні для прикладу функції/модулі, коли в останніх версіях дані елементи були перенесені або в інші модулі/бібліотеки або взагалі видалені.

Про ці відмінності між версіями можна було дізнатись на офіційному сайті бібліотеки. [10]

Висновки

Вподовж написання курсової роботи та реалізації її практичної частини було виконано ознайомлення з можливостями мови програмування Haskell для опису музичних творів.

Результатом роботи став практичний досвід в описі музики з використанням бібліотеки Euterpea та програми, що відтворюють описані в них музичні твори.

Дана робота розглядала лише частину можливостей мови Haskell та, більш конкретно, бібліотеки Euterpea для опису музики. Тому перспективою покращення результатів є більш глибоке вивчення нотного рівня бібліотеки та ознайомлення з сигнальний рівнем і більш тісною роботою з MIDI-файлами.

Список використаної літератури

1. Горбунова І. Б. Музыкальное программирование, или программирование музыки и музыкально-компьютерные технологии [Електронний ресурс] / Ірина Борисівна Горбунова // Теория и практика общественного развития. – 2015. – Режим доступу до ресурсу: http://teoria-practica.ru/rus/files/arhiv_zhurnala/2015/7/pedagogics/gorbunova.pdf.
2. https://www.youtube.com/watch?v=vt1PjMrJ2Yg
3. Orestis Melkonian. 2019. Music as language: putting probabilistic temporal graph grammars to good use. In Proceedings of the 7th ACM SIGPLAN International Workshop on Functional Art, Music, Modeling, and Design (FARM 2019). Association for Computing Machinery, New York, NY, USA, 1–10. DOI: https://doi.org/10.1145/3331543.3342576
4. Collins N. and D'Escrivan J., The Cambridge Companion to Electronic Music, New York, NY: Cambridge University Press, 2007.
5. Quick, Donya. (2015). Composing with Kulitta.
6. Quick, Donya & Morrison, Clayton. (2017). Composition by Conversation.
7. Hudak, Paul & Quick, Donya & Santolucito, Mark & Winograd-Cort, Daniel. (2015). Real-time interactive music in Haskell. 15-16. 10.1145/2808083.2808087.
8. Холомьев А. Учебник по Haskell [Електронний ресурс] / Антон Холомьев – Режим доступу до ресурсу: https://anton-k.github.io/ru-haskell-book/files/ru-haskell-book.pdf.
9. Szamozvancev, Dmitrij & Gale, Michael. (2017). Well-typed music does not sound wrong (experience report). ACM SIGPLAN Notices. 52. 99-104. 10.1145/3156695.3122964.
10. http://euterpea.com

11. Hudak P. (1996) Haskore music tutorial. In: Launchbury J., Meijer E., Sheard T. (eds) Advanced Functional Programming. AFP 1996. Lecture Notes in Computer Science, vol 1129. Springer, Berlin, Heidelberg
12. https://github.com/Euterpea/Euterpea2-Examples/tree/master/NoteLevel
13. Henning T. Audio Processing Using Haskell DFG-Schwerpunktprogramm 1114, Mathematical methods for time series analysis and digital image processing [Електронний ресурс] / Thielemann Henning // Zentrum für Technomathematik. – 2004. – Режим доступу до ресурсу: http://kurdish.parallelnetz.de/Research/haskellsignal.pdf
14. Юцевич Ю. Є. Вид. 2-ге, переробл. і доп. — Тернопіль: Навчальна книга - Богдан, 2009. — 352 с., 77 нотних прикладів та малюнків. — ISBN 978-966-10-0445-9
15. Signal Processing Methods for Music Transcription 2006th Edition by Anssi Klapuri (Editor), Manuel Davy (Editor)
16. Hudak, P., & Quick, D. (2018). The Haskell School of Music: From Signals to Symphonies. Cambridge: Cambridge University Press. doi:10.1017/9781108241861
17. https://github.com/MaciejWanat/Haskell-Euterpea-SevenNationArmy-tutorial