МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О. Є.
"_____"_____2021 р.

ДИПЛОМНИЙ ПРОЄКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ЗДОБУВАЧА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: «Інтерактивний додаток інтерпретації тональності та комбінацій тонів за *MIDI* вводом»

Виконавець: _____Саганюк М. Р.

Керівник: _____ Вавіленкова А. І.

Нормоконтролер: _____Тупота Є. В.

=

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет Кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання денна

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Литвиненко О. Є.
«_____»_____2021 р.

ЗАВДАННЯ
на виконання дипломної роботи (проєкту)

Саганюк Максим Русланович
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проєкту) Інтерактивний додаток інтерпретації тональності та комбінацій тонів за *MIDI* вводом»
затверджена наказом ректора від «_04_» лютого 2021 р. №__135/ст.
2. Термін виконання роботи (проєкту): з _17 травня 2021__ по _20 червня 2021___
3. Вихідні дані до роботи (проєкту): мова програмування *C++*, *QT Core*, структура музичних додатків, музична теорія, MIDI-повідомленя, MIDI-канал, MIDI-протокол
4. Зміст пояснювальної записки:
1) Особливості створення програмних музичних сервісі
2) Основні характеристики *MIDI* протоколу
3) Програмна реалізація інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом
5. Перелік обов'язкового графічного (ілюстративного) матеріалу:
1) Функціональна схема обробки голосових та режимних *MIDI* повідомлень
2) Діаграма послідовності застосування алгоритмів у інтерактивному додатку
3) Інтерфейс програми після введення даних з *MIDI* клавіатури
4) Схема алгоритму основного циклу програми

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Аналіз джерел за темою дослідження. Розроблення плану дипломного проекту	18.05.2021-21.05.2021	
2	Затвердження плану дипломного проекту та проведення консультацій з науковим керівником, щодо наповнення пояснювальної записки	22.05.2021-23.05.2021	
3	Дослідження систем відслідковування задач у проектах	24.05.2021-27.05.2021	
4	Розробка архітектури додатку для відслідковування задач	28.05.2021-31.05.2021	
5	Створення інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом	01.06.2021-09.06.2021	
6	Написання пояснювальної записки	10.06.2021-14.06.2021	
7	Підготовка демонстраційного матеріалу	15.06.2021 21.06.2021	

7. Дата видачі завдання: «_17_» _травня_ 2021 р.

Керівник дипломного проєкту _____ д.т.н., доц. Вавіленкова А.І._
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ _____Саганюк М.Р._____
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проєкту «Інтерактивний додаток інтерпретації тональності та комбінацій тонів за *MIDI* вводом» 59 с., 46 рис., 7 табл., 13 літературних джерел.

Об’єктом дослідження даного дипломного проєкту є процес автоматичного визначення тональностей.

Предметом дослідження є *MIDI*-протокол та *MIDI*-посилання.

Метою даного дипломного проєкту є проєктування та розробка інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом.

Методи дослідження – технології обробки *MIDI* інформації, математична модель дванадцяти тонового музичного строю, модель інтерпретації акордів на основі півтоноих відступів, алгоритм обробки поточних *MIDI*-повідомлень, алгоритм автоматичного пошуку акордів, алгоритм автоматичного визначення тональності, алгоритм автоматичного пошуку акордових інверсій.

Здійснено огляд існуючих музичних додатків; здійснено порівняльну характеристику функціональних можливостей цих додатків; проаналізовано структуру музичних додатків; здійснено огляд принципів музикальної теорії; реалізовано інтерактивний додаток інтерпретації тональності та комбінацій тонів за *MIDI* вводом.

Матеріали дипломного проєкту рекомендується використовувати при проведенні досліджень у сфері системного програмування та комп’ютетної інженерії з роботою *MIDI*-протоколу та *MIDI*-повідомлень, у допоміжних або навчальних цілях для музикантів.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – застосування в якості допоміжної системи для музикантів та в проєктах із застосуванням інформаційних технлогій у сферах, пов’язаних з музичною діяльністю.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ ТА ТЕРМІНІВ

MIDI - Musical Instrument Digital Interface (з англ. Цифровой интерфейс музыкальны

SMF - Standard MIDI File (з англ. Стандартний *MIDI* файл)

WAV – Waveform Audio File (з англ. в формі хвилі)

sus – sustainable (з англ. стійкий)

add – added (з. англ. доповнений)

m – minor (з англ. мінорний)

maj – major (з англ. мажорний)

mM – minor major (з англ. мінорно мажорний)

dim – diminished (з англ. зменьшений)

aug – augmented (з англ. збільшиний)

ВСТУП

Актуальність. Процес створення музики складається зі значної кількості складних процесів. Музикант повинен знати за якими правилами буде існувати його композиція. В якому музичному розмірі, темпі, тональності та ритмі вона буде написана. В залежності від яких акордів та акордових прогресій буде складатися композиція, залежить її якість та настрій. В процесі написання композиції, музикнат може відчути, що складний акорд який він використовує звучить добре не зразу зрозумівши що з себе цей акорд представляє в музикальній теорії, та в які з тональностей він входить. Це розуміння може допомогти музиканту зробити кращий вибір при переході від однієї тональності в іншу, може допомогти прийняти краще рішення для переходу до іншого акорду.

Даний дипломний проєкт – інструмент для музиканта, який допоможе йому краще розуміти свою композицію та робити більш якісні музичні рішення

Об’єктом дослідження даного дипломного проєкту є процес автоматичного визначення тональностей.

Предметом дослідження є *MIDI*-протокол та *MIDI*-посилання.

Метою даного дипломного проєкту є проєктування та розробка інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом з можливістю визначення акордових інверсій, акордових розширень та інших комплексних акордів

Методи дослідження дипломного проєкту: технології обробки *MIDI* інформації, за якою створена система обробки сигналів/повідомлень в реальному часі від *MIDI*-клавіатури до програми. Музична теорія використана для правильного та конкретного опису музичних процесів, для опису акордів, тональності та інших музичних правил. Математична модель дванадцяти тонового музичного строю для опису музичної теорії в

математичному контексті. Модель інтерпретації акордів на основі півтоноих відступів використана для створення системи за якою будуть працювати алгоритми інтерпретації *MIDI* сигналів. Алгоритм обробки поточних *MIDI*-повідомлень, алгоритм автоматичного пошуку акордів, алгоритм автоматичного визначення тональності, алгоритм автоматичного пошуку акордових інверсій використані як основа функціонування інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом.

Новизна дипломного проєкту полягає у проєктуванні швидкого, простого для розуміння та високоінтерактивного музичного додатку, який дозволяє в реальному часі визначати правильний акорд та тональність, під яку він підходить, та, на відміну від існуючих програмних музичних сервісів, які не містять динамічної функції розпізнавання акордів та тональності в процесі виконання композиції музикантом. Не залежно від складності акорду, додаток правильно визначає тональність, або визначає атональність акорду.

Практичне значення. У дипломному проєкті було створено інтерактивний додаток інтерпретації тональності та комбінацій тонів за *MIDI* вводом.

Особистий внесок випускника. Структура та алгоритми додатку були створені випускником особисто.

Практичні значення отриманих результатів в дипломному проєкті дають змогу використовувати отримані дані для порівняння з іншими музичними додатками. Використання результатів роботи додатку в комбінації з іншими додатками та як частину інформаційної технології.

Апробація отриманих результатів. Теоретичні аспекти отриманих у дипломному проєкті результатів проходили апробацію на міжнародній науково-технічній конференції “Інтелектуальні технології лінгвистичного аналізу”.

Публікації. Саганюк М.Р. Проблема методу підтвердження математичних істин в комп’ютерній інженерії: міжнар. науково-техн. конф.

«Інтелектуальні технології лінгвістичного аналізу», 20 – 21 жовтня 2020 р.: тези доп. – К., 2020. – С. 12.

Прогнозні припущення про розвиток об’єкту та предмету дослідження – застосування в якості допоміжної системи для музикантів та в проєктах із застосуванням інформаційних технлогій у сферах, пов’язаних з музичною діяльністю.

РОЗДІЛ 1
ОСОБЛИВОСТІ СТВОРЕННЯ ПРОГРАМНИХ МУЗИЧНИХ СЕРВІСІВ

1.1 Теоретичні основи створення програмних музичних сервісів

Програмні музичні сервіси можна поділити на дві основні категорії – сервіси для музикантів та сервіси для слухачів. Основні сервіси для слухачів – це музичні бібліотеки та стрімінгові сервіси, створені для легального прослуховування та купування музичних композицій, альбомів та інших продуктів музичного мистецтва.

Стрімінг (англ. *Streaming* - потоковий) - це спосіб передачі даних від провайдера до кінцевого користувача, при якому контент знаходиться на віддаленому сервері. Тут так само доречна аналогія зі звичайним теле- або радіомовленням

Стрімінгові (вони ж потокові) сервіси працюють за принципом передачі контенту від провайдера до користувача. Весь контент вже завантажений на сторонньому сервері, кінчений користувачеві не потрібно нічого завантажувати для перегляду або прослуховування. Контент транслюється в режимі реального часу, швидкість підвантаження безпосередньо залежить від швидкості інтернету користувача . З нинішнім навіть самим простеньким інтернетом можна без проблем прослуховувати музику та переглядати відео з стрімінгових сервісів.

Перегляд контенту онлайн став відмінною заміною скачування файлів. Ви не витрачаєте вільний простір на жорсткому диску (звичайно, якщо не забуваєте час від часу чистити тимчасові файли на пристрої), а відразу переглядаєте або слухаєте завантажувати. Такі сервіси потроху вбивають торренти, тому що все більше людей переходить на стрімінг з постійних закачувань. При цьому популярні сервіси легалізують перегляд всього того, що ви хочете, тому що ви просто оплачуєте підписку на певний час. Виходить той же телебачення (або радіо), але з більш розширеним функціоналом і відсутністю запланованої програми передач з обов'язковими рекламними паузами.

Прикладами музичних бібліотек/музичних стрімінгових сервісів є:

- *Spotify* (Створена *Spotify Ltd*.)
- *Apple Music* (Створена *Apple Inc*.)
- *Google Play Music* (Створена *Google LLC*)
- Яндекс.Музика (Створена *Yandex Apps*)
- *Deezer* (Створена *Deezer Moblile*)
- *Bandcamp* (Створена *Bandcamp Inc*.)
- *SoundCloud* (Створена *Soundcloud*)
- *Youtube Music* (Створена *Google LLC*)

Проведено порівняльну характеристику популярних стрімінгових сервісів за такими критеріями, як: вартість підписки, пробний період, кількість пісень, наявність офлайн режиму, наявність радіо та текстів пісень (табл. 1.1.). [3]

Таблиця 1.1

Порівняльна характеристика можливостей стрімінгових сервісів

	VK	СберЗвук	Яндекс	*Apple*	*Deezer*	*Youtube*	*Spotify*
Вартість підписки	≈55 грн/міс	≈60 грн/міс	≈60 грн/міс	≈60 грн/міс	≈60 грн/міс	≈75 грн/міс	≈60 грн/міс

Продовження таблиці 1.1

Пробний період	1 місяць	1 місяць	3 місяці	3 місяці	1 місяць	1 місяць	3 місяці
Пісень	-	≈25 млн	≈60 млн	≈70 млн	≈56 млн	-	≈50 млн
Тексти пісень	Частково	Так	Так	Так	Так	Ні	Частково
Оффлайн режим	Так	Так	Так	Так	Так	Так	Так
Загрузка своєї музики	Ні	Ні	Так	Так	Так	Так	Так
Радіо	Так	Ні	Так	Так	Ні	Так	Так

Проведено порівняльну характеристику популярних стрімінгових сервісів за такими критеріями, як: простота інтерфейсу, швидкість роботи, звучність використання та інтеграція з пристроями за 5ти бальною шкалою (табл. 1.2).

Таблиця 1.2

Порівняльну характеристику популярних стрімінгових сервісів

	VK	СберЗвук	Яндекс	*Apple*	*Deezer*	*Youtube*	*Spotify*
Проста інтерфейсу	3	4	5	5	5	5	5
Швидкість роботи	4	4	5	4	5	5	5
Зручність використання	3	3	5	3	5	3	5

Продовження таблиці 1.2

Інтеграція з пристроями	4	4	5	3	5	5	5
Середня оцінка	3,5	3,75	5	3,75	5	4,5	5

Стрімінгові музичні сервіси використовують технологію потокових мультимедіа. Потокова мультимедіа – це мультимедіа, яка безперервно отримується користувачем від провайдера потокового мовлення через Інтернет.

Для слухачів також існують сервіси інших типів та задач. Наприклад, сервіс *Shazam* не є музичною бібліотекою. Цей сервіс дозволяє визначити назву та виконавця композиції за допомогою запису фрагменту композиції мікрофоном. Сервіс порівнює фрагмент з центральною базою даних та при успішному порівнянні видає інформацію. Також музичним сервісом для слухачів є музичні бази метаданих.

Музичні бази метаданих – бази даних музичних композицій та альбомів, які детально описують музичну композицію/альбом за жанром, датою випуску та запису, жанрами та піджанрами, дескрипторами. Прикладами таких сервісів є *rateyourmusic.com* та *last.fm*.

Програмні сервіси для музикантів – програмні сервіси, що допомагають музикантам у виробництві музики. Основні сервіси для музикантів можна поділити на робочі станції, допоміжні сервіси та навчальні. Навчальні та допоміжні відрізняються основною аудиторією, на яку направлені ці сервіси. Допоміжні будуть корисні музикантам не залежно від досвіду, навчальні будуть корисні лише початковим музикантам. Але допоміжні та навчальні функції можуть бути в одному програмному сервісі.

Робочі станції для музикантів (цифрові звукові робочі станції та сіквенцери (*sequencer*)) - електрона та/або комп’ютерна система призначена для запису та редагування цифрового звуку, зазвичай використовують протокол *MIDI*.

Прикладами робочих станцій для музикантів є

FL Studio (Створена *Image-Line Software*) — редактор-секвенсер для написання музики, створений 1997 року програмістом Дідьє Дембрено (рис. 1.1).

Рис. 1.1. Приклад роботи *FL Studio*

Музика створюється шляхом запису і зведення (звукозапису) аудіо-, або *MIDI*-матеріалу. Готова композиція може бути записана у файл з розширенням *WAV*, *MP*3 або *OGG*. Програма написана мовою програмування *Delphi*

Ableton Live (Створена *Ableton*) — програмне забезпечення для музикантів та діджеїв, засноване на техніці звукових петель (рис. 1.2).

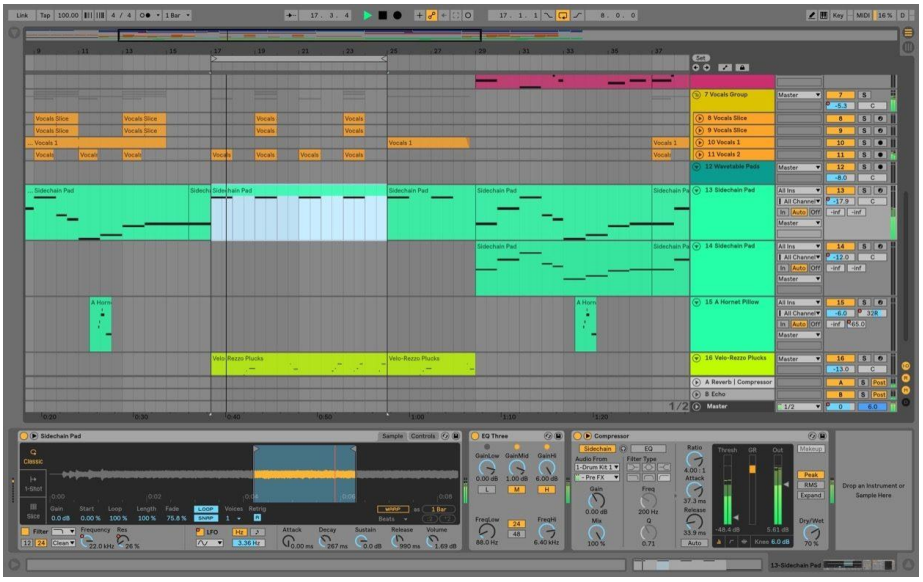

Рис. 1.2. Приклад роботи *Ableton Live*

Програма має два режими — для створення музики в класичному студійному режимі («*Arrangement View*»), та реальному часі («*Session View*»).

GarageBand (Створена *Apple Inc*.) — непрофесійна програма для створення музичних композицій (рис. 1.3).

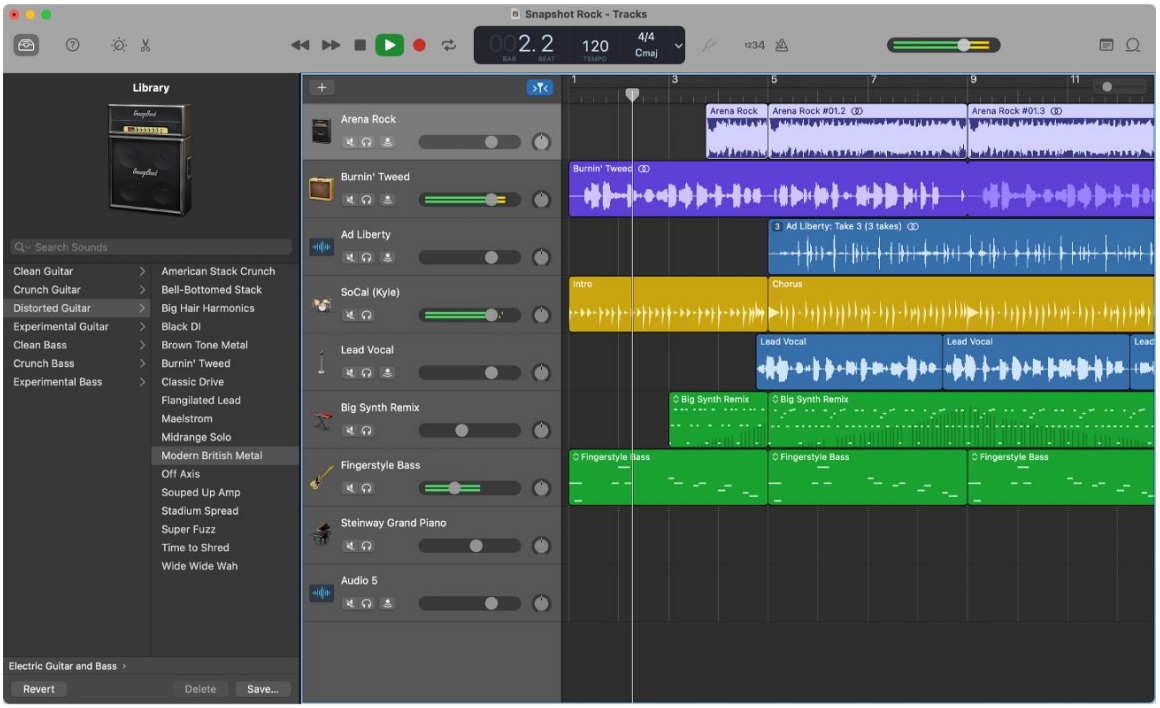

Рис. 1.3. Приклад роботи *GarageBand*

Серед можливостей: запис та імпорт *MIDI*, велика бібліотека власних музичних інструментів, бібліотеки готових семплів, можливість експорту та обміту з іншими пакетами *iLife*.

Допоміжними сервісами для музикантів є тюнери. Тюнери за допомогою мікрофону визначають, яку саме частоту створює струна та чи співпадає ця частота з необхідною частотою для ноти в дванадцяти тоновому рівномірно-темперованому строю. Прикладами таких сервісів/програм є – *Guitar Tuna. [4]* (рис. 1.4)

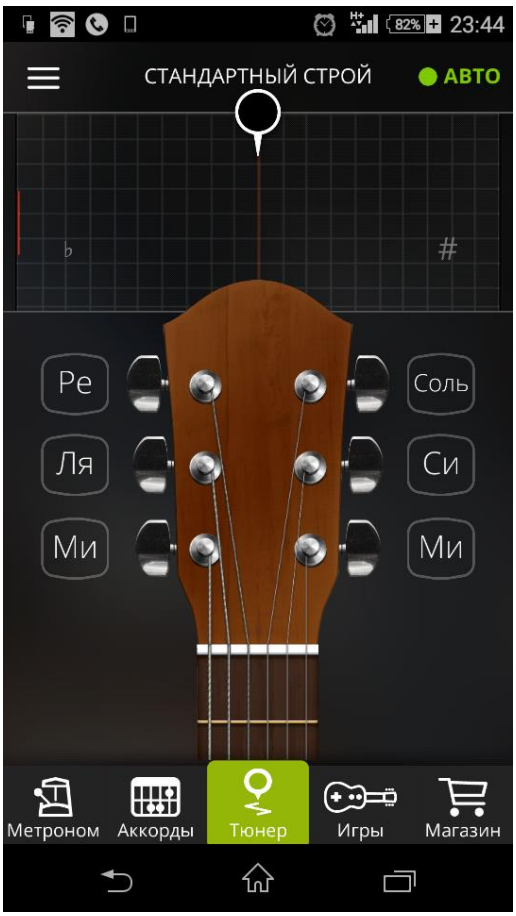

Рис. 1.4. Скріншот роботи *GuitarTuna*

Навчальними сервісами для музикантів є, наприклад, *EarMaster.School* створений для тренування музичного слуху.

Дана дипломна робота ставить за ціль створення допоміжної програми для джаз-піано музикантів (але і не обмежена ними).

1.2. Порівняльна характеристика існуючих програмних музичних сервісів

Більшість існуючих програм для музикантів, які мають функцію розпізнавання тональності, використовують існуючий аудиофайл, який вони аналізують. Іншими словами, більшість програм використовують статичний аналіз аудіо потоку. Основна ідея – створити динамічний алгоритм розпізнавання тональності, де вхідні дані – це процес гри на музичному інструменті, в даному випадку – клавішному.

Також програма повинна буде динамічно визначати акорд (ритмічно одночасне сполучення кількох (не менше трьох) різних за висотою звуків). По кількістю звуків, програма повинна буде визначати прості тріади, але також:

– септакорди (4-х нотні акорди)
– нонакорди (5-х нотні акорди)
– ундецімкорди (6 нотні акорди)
– терцдецимакорди (7 нотні акорди).

А також повинна визначати акордові інверсії, *sus*-акорди, *add*-акорди, акордові інверсії, тобто комплексні акорди. Також це

Прикладами програм, які виконують схожу, але іншу задачу є такі програми:

– *Mixed in Key*
– *EarMaster.School*
– *goodEar Pro*

Mixed in Key – допоміжна програма для музикантів, яка визначає тональність та іншу корисну для музиканта за вхідним цифровим файлом типу *mp*3. Створена для персональних комп'ютерів *[5]* (рис. 1.5).

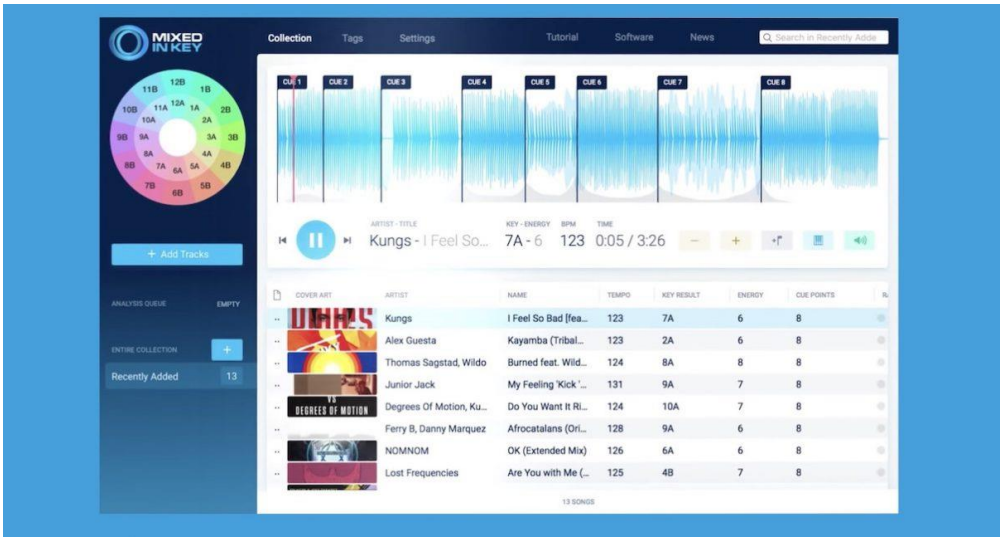

Рис. 1.5. Скріншот роботи *Mixed in Key*

Earmaster.School – навчальна програма, що призначена для початкових музикантів. Інтерактивно навчає музикальним концептам такі як ноти, акорди, інтервали та іншим. Створена для персональних комп'ютерів (Рис. 1.6)

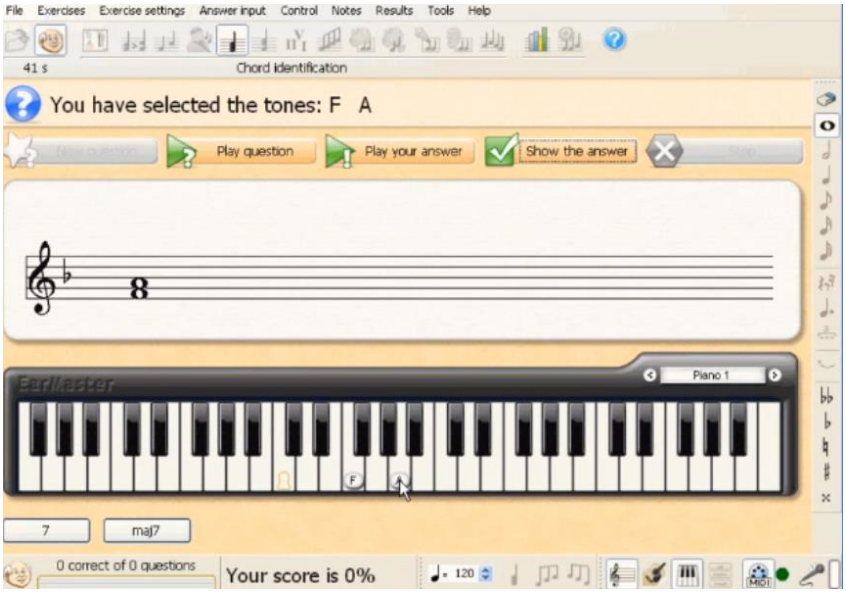

Рис. 1.6. Скріншот роботи *Earmaster.School*

goodEar Pro – навчальна програма, що призначена для початкових музикантів. Інтерактивно навчає музикальним концептам такі як ноти, акорди, інтервали та іншим. Створена для смартфонів (рис. 1.7)

Рис. 1.7. Скріншот роботи *goodEar Pro*

Розглянемо основні характеристики розглянутих вище музичних сервісів та визначимо їх переваги та недоліки (табл.. 1.3).

Таблиця 1.3

Основні характеристики розглянутих вище музичних сервісів

	Mixed in Key	*Earmaster.School*	*goodEar Pro*
Вхідні дані	Аудіо файл типу *mp*3	*MIDI*-потік з піано-клавіатури або аудіо інформація з мікрофону	Піано-клавітура на екрані
Визначення тональності	Так	Частково	Ні
Визначення модальності	Ні	Ні	Ні
Визначення назви простих акордів	Ні	Так	Так
Визначення назви комплексних акордів	Ні	Частково	Так
Платформа	Персональний комп’ютер	Персональний комп’ютер	Персональний комп’ютер

Вищевказані програмні музичні допоміжні сервіси для музикантів не пропонують динамічної функції розпізнавання акордів та тональності в процесі виконання композиції музикантом. Також вищевказані програмні сервіси не пропонують функціоналу для визначення модальності. Деякі приклади мають функцію визначення простих акордів, але не комплексних.

Виявлені недоліки дають змогу сформувати вимоги до створення нового інтерактивного додатку для чого необхідно:

- ознайомитися з особливостями створення програмних музичних сервісів;

- проаналізувати існуючі програмні музичні сервіси та здійснити їх порівняльну характеристику;

- дослідити технології та алгоритми обробки звукової інформації;

- вивчити алгоритм інтерпретації тональності та комбінацій тонів;

- створити інтерактивний додаток інтерпретації тональності та комбінацій тонів за *MIDI* вводом.

1.3 Висновки до розділу

В першому розділі було описано та досліджено існуючі сервіси для музикантів та слухачів музики. Між сервісами для слухачів була здійснена порівняльна характеристика за такими параметрами як швидкодія, звучність та простота інтерфейсу. Були виділені деякі найкращі сервіси з них за п’яти бальною шкалою. Була здійснена порівняльна характеристика між існуючими музичними сервісами для музикантів та визначено, що вони не пропонують динамічної функції розпізнавання акордів та тональності в процесі виконання композиції музикантом, а також не надають функціоналу для визначення модальності. Також визначено, що існуючі сервіси для музикантів не пропонують функціоналу для динамічного визначення всіх можливих тональностей для акордів та для динамічного визначення комплексних акордів.

РОЗДІЛ 2
ОСНОВНІ ХАРАКТЕРИСТИКИ *MIDI* ПРОТОКОЛУ

2.1. Формат, комутація та канали *MIDI* протоколу

MIDI - це протокол зв'язку між пристроєм управління, генеруючим команди, і підлеглим пристроєм, який виконує ці команди. *MIDI* дозволяє виконавцю натиснути клавішу на одному інструменті, а отримати при цьому звук іншого або навіть декількох. Будь-які дії виконавця на органи управління (натискання клавіш, педалей, зміна положень регуляторів і т. п.) можуть бути перетворені в команди, які можна передати по *MIDI*-кабелю на інші інструменти. Ці інструменти, отримуючи команди, обробляють їх так само, як і при впливі на їх власні органи управління.[6]

Інтерфейс дозволяє одноманітно кодувати в цифровій формі такі дані як натискання клавіш, настройку гучності та інших акустичних параметрів, вибір тембру, темпу, тональності і ін., з точною прив'язкою в часі. В системі кодувань присутня безліч вільних команд, які виробники, програмісти і користувачі можуть використовувати на свій розсуд. Тому інтерфейс *MIDI* дозволяє, крім виконання музики, синхронізувати управління іншим обладнанням, наприклад, освітлювальним

Послідовність *MIDI*-команд може бути записана на будь-який цифровий носій у вигляді файлу, передана по будь-яких каналах зв'язку. Відтворює пристрій або програма називається синтезатором (секвенсором) *MIDI* і фактично є автоматичним музичним інструментом. [7, с. 100]

Стандартний *MIDI*-файл (*SMF - Standard MIDI File*) - це спеціально розроблений формат файлів, призначений для зберігання даних, що записуються і / або виконуваних секвенсором, секвенсор може бути як програмою для комп'ютера, так і апаратно виконаним модулем.

На відміну від інших форматів, це не оцифрований звук, а набори команд (програються ноти, посилання на які відтворюється інструментами, значення змінних параметрів звуку), які можуть відтворюватися по-різному в залежності від пристрою відтворення. Зручність формату *MIDI* як формату представлення даних дозволяє реалізовувати пристрої, що здійснюють автоматичне аранжування за заданими акордами.

У цьому форматі зберігаються стандартні *MIDI*-повідомлення (тобто статус-байти і відповідні їм байти даних), а також тимчасові мітки або маркери для кожного повідомлення (тобто послідовності байтів, які вказують, скільки умовних одиниць часу необхідно виконувати ноту або паузу).

При розробці формату передбачається можливість для будь-якого секвенсора читати і записувати файл таким чином, щоб, з одного боку, не загубилися його дані, а з іншого боку, щоб формат був досить гнучким, додатки могли зберігати в файлах свою специфічну інформацію, зрозумілу тільки їм, але не зрозумілу іншим програмам-додатків, причому при завантаженні файлів *MIDI* незрозуміла іншим програмам-додаткам.

Комутація *MIDI* виконується наступним чином. У нас є два синтезатора, і ми хочемо, щоб при натисканні клавіші на одному з них другий синтезатор зіграв ту ж ноту, але своїм звуком. Очевидно, для цього потрібно зробити на першому синтезаторі вихідний *MIDI*-роз'єм, а на другому - вхідний *MIDI*-роз'єм і з'єднати інструменти *MIDI*-кабелем. Перший синтезатор при натисканні клавіші повинен генерувати повідомлення про взяття ноти і посилати його на свій вихід, а другий синтезатор - отримувати це повідомлення через вхід і відтворювати звук [6] (рис 2.1)

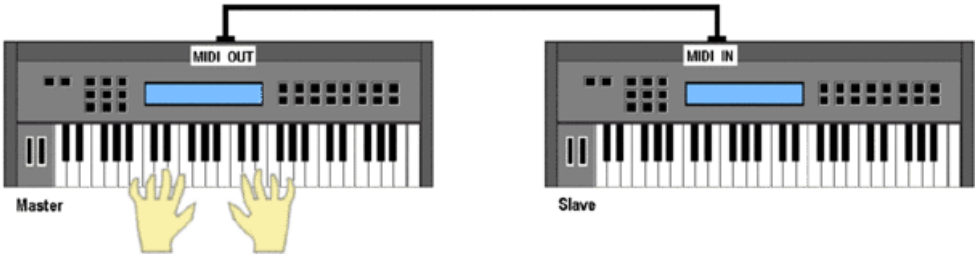

Рис. 2.1. Комутація двох *MIDI* пристроїв

Сучасні *MIDI*-пристрої мають три роз'єми — вхідний роз'єм *MIDI-IN*, через який пристрій отримує повідомлення ззовні та два вихідних роз'єми — *MIDI-OUT* та *MIDI-THRU*. Різниця між портами *MIDI-OUT* та *MIDI-THRU* полягає в тому, що сигнал з *MIDI-OUT* генерується безпосередньо на самому інструменті, тоді як порт *MIDI-THRU* видає точну копію сигналу, який отримує пристрій через вхідний порт *MIDI-IN*.

MIDI - протокол цифровий, і висота нот кодується числами, а не напругою. Цифри передаються по *MIDI* кабелю у вигляді дуже коротких імпульсів.

MIDI використовує послідовний тип інтерфейсу. Послідовний інтерфейс означає, що імпульси по *MIDI*-кабелю передаються один за іншим. Так що в кожен момент часу приймача досягає лише один імпульс. Значить, передати одночасно кілька повідомлень через один *MIDI*-роз'єм неможливо. *[6]*

Перш за все, кожен музичний звук містить в своєму складі основний тон, який представляє собою гармоніку максимальної амплітуди, яка і визначає ноту що програється.

Крім основного тону музичний звук також містить сторонні гармоніки, що носять назву обертонів. Існування сторонніх гармонік породжує забарвлення звуку і формує обертональний портрет, який різний у кожного інструменту. Завдяки цьому на основі зразка грається звуку стає можливим визначення типу інструменту. (рис 2.2)

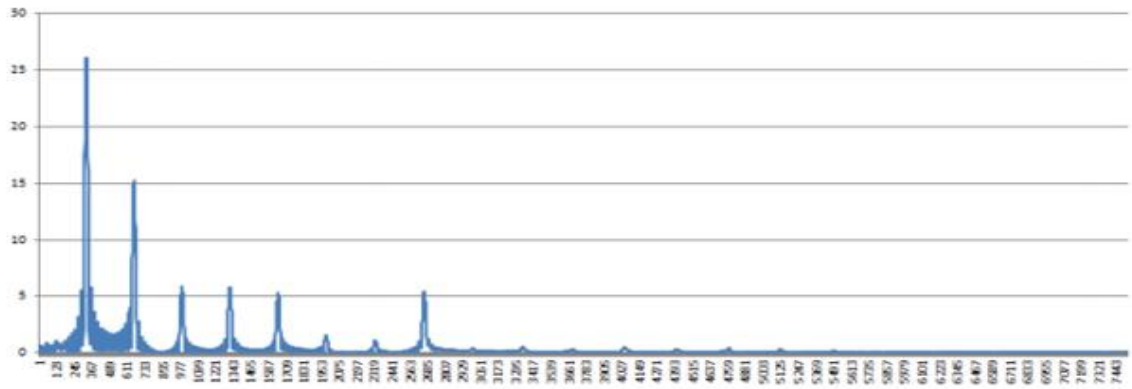

Рис. 2.2. Спектр звуку піано

Всі звуки, які представлені на цих малюнках, мають одну і ту ж частоту основного тону, яка представлена як пік з найбільшою амплітудою, проте відносна конфігурація інших піків, є обертональними гармоніками, відмінні для кожного випадку.

Також необхідно відзначити, що дослідження спектра музичних звуків, в тому числі представлених на вищенаведених малюнку, показує, що, як правило, обертони слідують на частотах, кратних частоті основного тону.

Перш за все, необхідно ввести модель музичної події, яка могла б здійснювати подання програмуючих музичних подій. Його можна визначити як кортеж

$$E = < N, t_{\text{Н}}, t_{\text{Д}}, I >$$

де N-код ноти, $t_{\text{Н}}$ - час початку програвання ноти, $t_{\text{Д}}$ - тривалість програвання, I - тип інструменту.

Символьне представлення музичного твору - множина подій

$$M = \{E_1, E_2, E_3, \dots, E_n\},$$

де E - події, що становлять музичний твір.

В якості представлення, придатного для перетворення в класичний вид можна уявити модель, аналогічну перетворенням в класичний вид, проте включаючи в себе позначення висоти ноти і то її тривалість в долях такту.

Таким чином, класичну ноту можна уявити як кортеж

$$N = < N, t_{\text{Н}}, t_{\text{Д}} >,$$

виключивши з розгляду тип інструменту, так як зазвичай тип інструменту визначається для всієї партії. Таким чином, також доцільно ввести наступну модель для партій. партія буде представляти із себе також кортеж

$$P = \{N_1, N_2, N_3, \dots, N_n\}$$

а весь музичний твір матиме вигляд множини

$$\text{C} = \{P_1, P_2, P_3, \dots, P_n\}$$

де P – це окрема партія.

Також вважається за необхідне формування моделі вхідного сигналу. В якості вхідних даних розробляється програмна система яка використовує аудіофайли формату *WAV*, аудіо дані яких являють собою оцифрований звуковий сигнал в імпульсно-кодової модуляції. Таким чином, вхідний сигнал *S* можливо уявити як упорядкований масив

$$S = \{S_1, S_2, S_3, \dots, S_n\}$$

де S_i - значення *i*-го відліку.

В якості моделі спектрограми можливо вибрати модель матриці, в якій номер стовпчика відповідає номеру фрейму, а номер строки номеру гармоніки. Таким чином, якщо спектрограмма містить n відліків, і m гармонік, то її модель можна представити таким чином:

$$G = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1m} \\ g_{21} & g_{22} & \cdots & g_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nm} \end{pmatrix}$$

Використовуються 12-мірні шаблонні вектори для всіх вживаних в музиці поєднань з трьох нот: мажорних, мінорних, збільшених і зменшених тризвуків. Мажорні і мінорні тризвуки застосовуються значно частіше, ніж збільшені і зменшені. Шаблони для акордів, що складаються з 4 і більше нот, не використовуються врамках даного методу, а значить, результатом розпізнавання такого акорду буде одне з тризвуків. Наприклад, шаблон для акорду до мажор має вигляд (1.3, 0, 0, 0, 1, 0, 0, 1.3, 0, 0, 0, 0) [8, с. 12]

При грі акордів ноти кожного акорду будуть передаватися послідовно. Фактично, замість акордів синтезатор-приймач буде виконувати тільки дуже-дуже щільне арпеджіо.

Для кожного акорду можна визначити відповідний йому реальний вектор, що представляє собою 12-мірний вектор *V*2., що складається з послідовності 0 та 1, де

$$V_z[i] = \begin{cases} 1, \text{якщо } i\text{та нота 12ти тоновому строю } \epsilon \text{ акорд} \\ 0, \text{в іньшому випадку} \end{cases}$$

Так, наприклад, акорд C*maj* який складається з нот до мі та соль можно зіставити в наступний вектор [9, с. 11]

$$V_{Cmaj} = (1,0,0,0,1,0,0,1,0,0,0,0)$$

Отже, ми з'єднали два синтезатора *MIDI*-кабелем, по якому передаються повідомлення про взяття нот (в тому числі і декількох одночасно, з поправкою на попереднє зауваження). Тепер ми хочемо підключити ще два синтезатора, але управляти ними як і раніше з першого. Для цього можна зробити в першому синтезаторі кілька *MIDI*-виходів і з'єднати їх трьома *MIDI*-кабелями з *MIDI*-входами інших, тобто, кажучи комп'ютерною мовою, організувати мережу з топологією "зірка" [6] (рис 2.3)

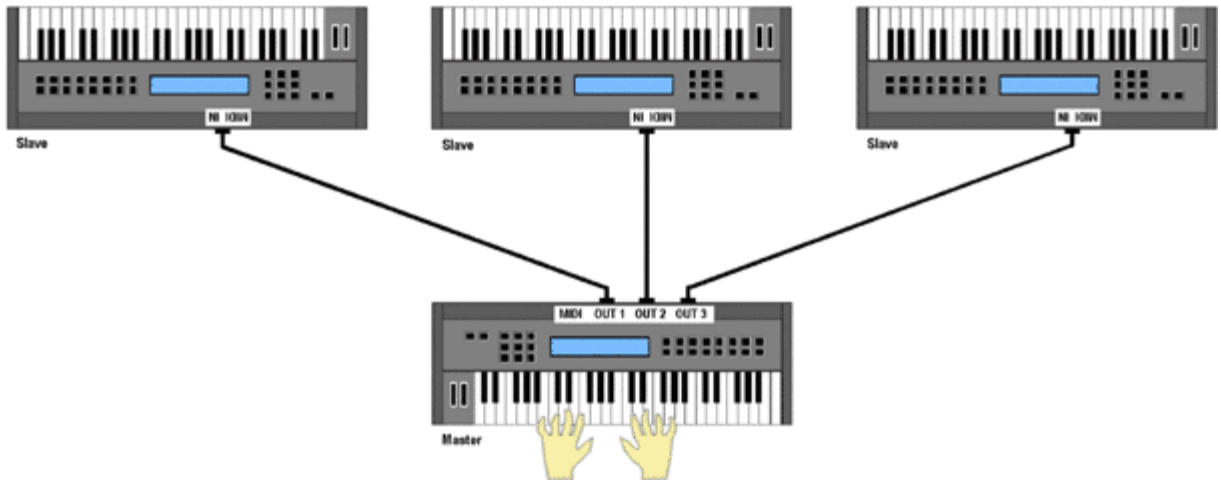

Рис. 2.3. Комутація чотирьох *MIDI* пристроїв за топологією “зірка”

Це цілком можливо, але не всі інструменти (хоча б з економічних міркувань) будуть обладнуватися декількома *MIDI*-виходами. А якщо буде потрібно управляти ще двома або трьома синтезаторами, то виходів точно не напасешся. Було знайдено рішення: зробити додатковий роз'єм *MIDI Thru* (наскрізний). Завдання синтезатора, обладнаного таким роз'ємом - дублювати

на нього всі повідомлення, що приходять на вхід *MIDI In*. Тоді нашу систему з чотирьох синтезаторів можна з'єднати ланцюжком (рис 2.4)

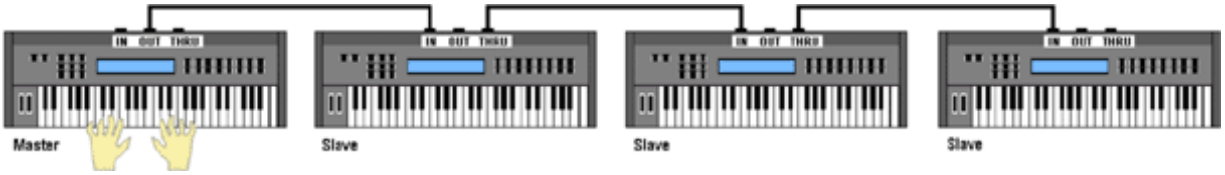

Рис. 2.4. Комутація чотирьох *MIDI* пристроїв послідовно за допомогою *MIDI Thru*

Якщо необхідно передавати інформацію не лише про ноту в часі, а і про деякий інструмент можна використовувати канали.

MIDI сигнал можна передавати по шістнадцяти логічним каналам. Слово "логічний" означає те, що всі канали існують у вигляді абстракції і передаються по одному *MIDI*-кабелю. Просто кожне повідомлення, наприклад, про взяття ноти, забезпечується додатковим числом - номером каналу, на якому звучить нота. У цьому докорінна відмінність *MIDI* від аналогової звукової комутації, де, наприклад, партія рояля йде з магнітофона на мікшер за окремим кабелем, який можна помацати руками. В *MIDI* по одному кабелю можуть передаватися одночасно партії всього симфонічного оркестру (природно, у вигляді керуючих команд, а не звуку).

Якщо перший синтезатор дозволяє розділити клавіатуру на дві зони, то можна призначити зону для лівої руки на передачу по каналу 8, а для правої - по каналу 4. Тоді ми зможемо одночасно виконувати партії баса і рояля. При цьому на першому синтезаторі буде потрібно тільки один *MIDI*-вихід (рис 2.5)

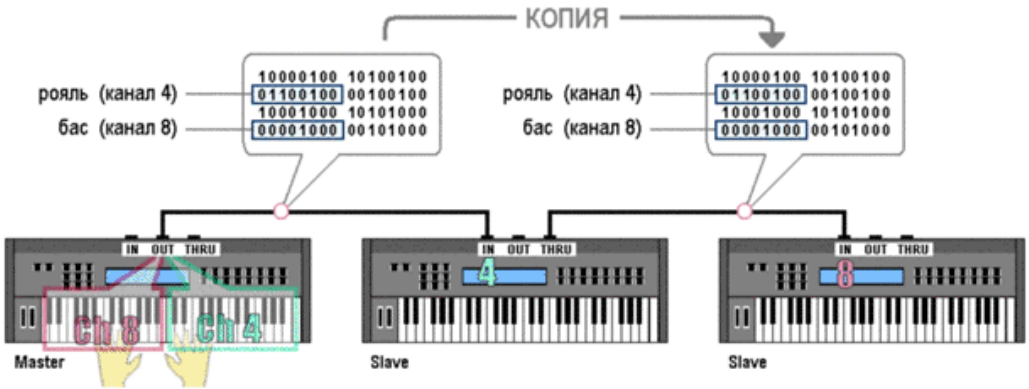

Рис. 2.5. Передача *MIDI* сигналу використовуючи два канали

2.2 Мова *MIDI* та структура основних повідомлень

MIDI-повідомлення - це потік даних в реальному часі. При передачі даних іноді можуть відбуватися їх втрати та інші неприємності. У комп'ютерних мережевих протоколах, в разі приходу зіпсованих даних (що перевіряється за контрольною сумою) відбувається повторний запит до сервера до тих пір, поки дані не прийдуть в цілості й схоронності. У протоколі *MIDI* така можливість відсутня.

Мова *MIDI* складається тільки з команд управління і параметрів цих команд. Нічого іншого по *MIDI*-кабелю не передається.

Повідомлення логічно розділити на два основних типи: одні керують звукоутворенням, тобто говорять, наприклад, яку ноту і як голосно грати, другі виконують службові функції, на зразок зміни налаштувань тон-генератора і синхронізації. Оскільки вилучення звуків відбувається в *MIDI* на певному каналі, повідомлення першого типу називаються повідомленнями каналу (*Channel Messages*).

Повідомлення каналу діляться, в свою чергу, на голосові (*Channel Voice Messages*) і повідомлення режиму каналу (*Channel Mode Messages*). Системні повідомлення діляться на загальносистемні (*System Common Messages*), повідомлення реального часу (*System Real Time Messages*) і ексклюзивні (*System Exclusive Messages*) *[7, с*. 104*]* (рис. 2.6)

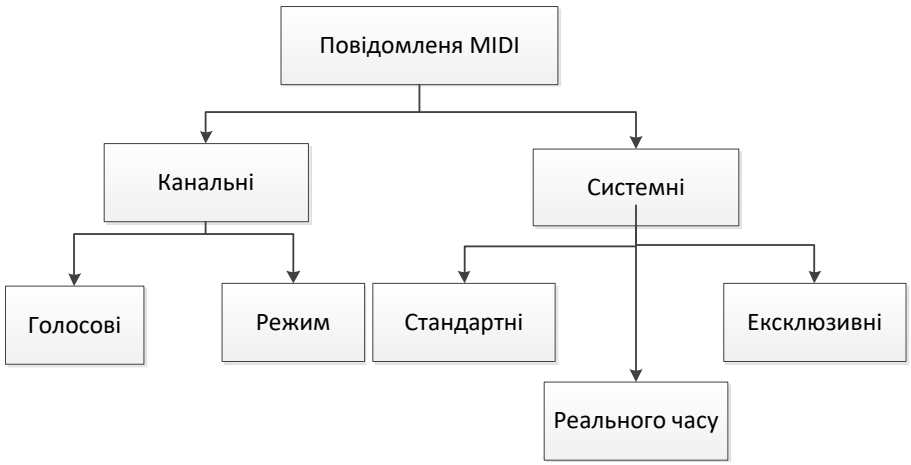

Рис. 2.6. Типи повідомлень

Голосові повідомлення каналу – повідомлення, які посилають в тон-генератор інформацію про управління звуком. Вони "доповідають" тон-генератору про те, що зараз робить виконавець - натискає кнопку, крутить колесо модуляції, рухає фейдер або відпускає педаль. Тобто голосові повідомлення описують дії виконавця в цифровій формі. [7, с. 105]

Загальносистемні повідомлення виконують кілька різнорідних завдань. У їх числі - синхронізація *MIDI*- і аудіопристроїв за допомогою протоколу *MIDI Time Code* (*MTC*), передача позиції пісні, вибір пісні і навіть запит на підстроювання осциляторів синтезатора.

Системні повідомлення реального часу призначені для синхронізації *MIDI*-пристроїв, наприклад, секвенсоров і драм-машин, по протоколу *MIDI Clock*.

Системні ексклюзивні повідомлення (що позначаються для стислості *SysEx*) – специфічний протокол повідомлень для виробників пристроїв

Спочатку розглянемо системні повідомлення.

У статусному байті системних повідомлень 4 старші байти мають значення «1», 4 молодші визначають тип системного повідомлення. Системні повідомлення слугують головним чином для керування роботи програмою, вони не поділяються на канали і стосується всієї системи і мають вищий пріоритет, ніж канальні.

Усього передбачено 16 системних повідомлень, проте використовуються з них лише 11 [10] (табл. 2.1):

Таблиця 2.1

Системні повідомлення MIDI

№	Назва	Призначення
*F*0	*System Exclusive*	Початок *SysEx*
*F*1	-	-

F2	*Song Position Pointer*	Дана пісня

Продовження таблиці 2.1

F3	*Song Select*	Вибір пісні
F4	-	-
F5	-	-
F6	*Tune Request*	Запит підстроювання
F7	*EOX*	Кінець *SysEx*
F8	*Timing Clock*	Синхронізація за часом
F9	-	-
FA	*Start*	Початок гри на партитурі
FB	*Continue*	Продовження гри на партитурі
FC	*Stop*	Зупинка гри на партитурі
FD	-	-
FE	*Active Sensing*	Перевірка з'єднань
FF	*System Request*	Скидання всіх пристроїв мережі

Тепер розглянемо голосові повідомлення основного каналу (табл.2.2):

Таблиця 2.2

Голосові повідомлення MIDI

Код	Повідомлення	Значення	Дапазон
1001	*Note-on*	Натиснення клавіші	0-127 Номер клавіші 0-127

			Сила натиснення

Продовження таблиці 2.2

1000	*Note-off*	Відпускання клавіші	0-127 Номер клавіші
1100	*Program Change*	Включення інструменту	0-127 Номер інструменту
1110	*Pitch Bend*	Зсув висоти звуку	-8191-8191 Величина зсуву
1011	*Control Change*	Зміна контролеру	0-127 Номер контролеру Та значення контролеру
1010	*Key Pressure*	Тиск на клавішу	0-127 Сила тиску на клавішу
1101	*Channel Pressure*	Тиск на клавіатуру	0-127 Сила тиску на клавішу

Алгоритм інтерпретації комбінацій тонів за *MIDI* вводом представлено на рис. 2.6.

Коли натискається клавіша, надсилається повідомлення *note-on*. Це повідомлення складається з двох частин: перша-висота (*note*) і як швидко її натиснули (*velocity*). «*Note*» описує висоту (*pitch*, пітч) зі значенням між 0 та 127. «*Velocity*» теж має значення 0-127 і зазвичай описує гучність (*gain*). Чим

більша *velocity*, тим гучніше. Інколи різна швидкість натиснення створює різний тембр. Або впливає на швидкість атаки звуку. Чим швидше натискаєш, тим різкіше починає звучати звук. Ці параметри залежать від того як налаштований звук на синтезаторі. Також не всі електронні клавіатури мають чутливість до швидкості натискання. Комп'ютерна клавіатура теж не має такої чутливості, тому коли її використовувати як джерело міді-повідомлень (за допомогою спеціальних програм, наприклад *MIDI YOKE*), то секвенсер буде отримувати одне і теж саме значення *velocity*.

Коли говорять: "Ця клавіатура - активна", зазвичай мають на увазі, що гучність звуку залежить від сили натискання на клавішу, як у фортепіано. Насправді такі клавіатури реагують не на "силу", а на швидкість натискання. Реакція ж на силу натискання називається *aftertouch* - "післядотиком" (в англомовній технічній літературі застосовується також термін "*pressure*" - тиск), і цією здатністю володіють далеко не всі клавіатури [11]

Коли клавіша відпускається, то створюється інше повідомлення *note-off*. Це повідомлення теж має частину «*note*», щоб переконатись, що повідомлення про закінчення звучання відноситься до потрібної ноти. Наприклад, якщо натиснути дві ноти, і відпустити одну, то друга продовжить звучання. Інколи *note-off* теж може містити повідомлення *velocity*, що надасть синтезатору інформацію про те, як закінчити звук. (рис 2.7)

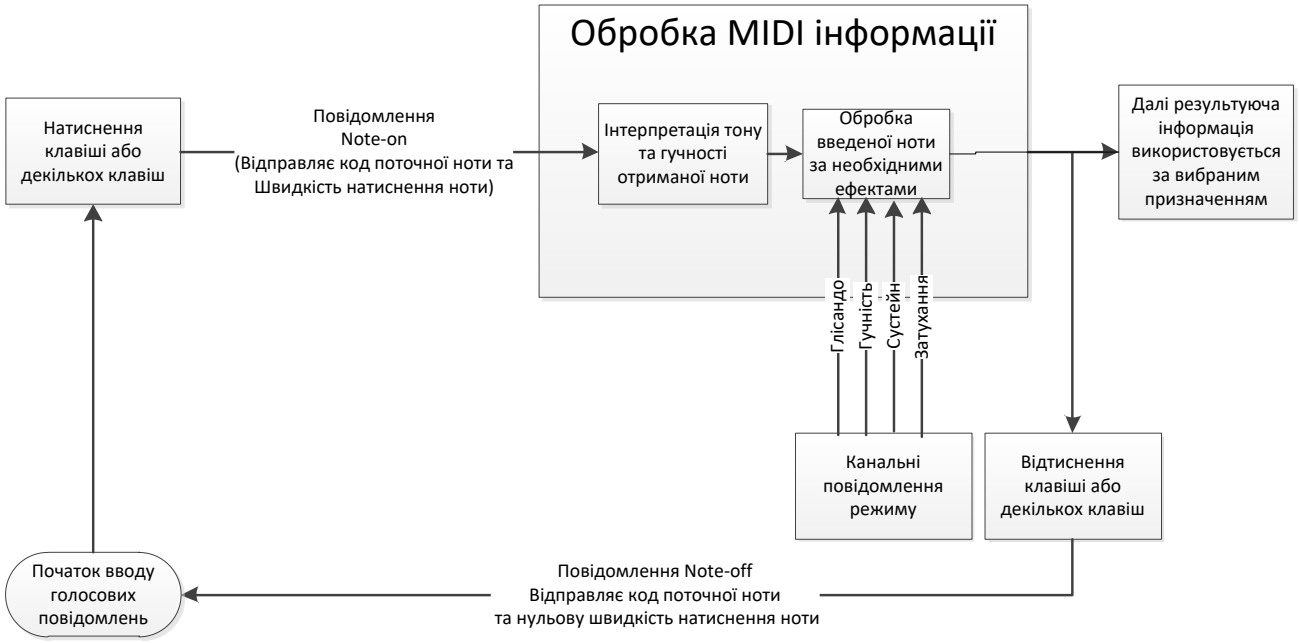

Рис. 2.7. Функціональна схема обробки голосових та режимних *MIDI* повідомленя

Спосіб використання цих повідомлень може бути різниЬм в залежності від виду *MIDI* апаратури (табл. 2.3).

Таблиця 2.3

№	Назва	Значення	Дапазон
1	*Modulation*	амплітудне вібрато	0-127 Мін та макс вібратор
5	*Portamento*	Час глісандування (працює при включенному режимі глісандування (контролер № 65)	0-127 Мін та макс значення
7	*Volume*	гучність звуку	0-127 Мін та макс гучність
10	*Pan*	просторова локалізація звуку	0-127 0 – локалізація зліва

			64 – локалізація по центру 127 = локалізація справа
11	*Expression*	Виразність виконання (як правило діє аналогічно контролеру № 7)	0-127 Мін та мккс значення
64	*Sustain*	Затримка звуку (Ефект, аналогічний правій педалі фортепіано)	0-127 Мін та макс значення

Загальне повідомлення можна записати у вигляді комбінації декількох мікроповідомлень, а саме:

- Повідомлення про синхронізацію
- Повідомлення про вибрану клавішу
- Повідомлення про інтенсивність натиснення клавіші
- Інші службові повідомлення

Де кожне роздільне повідомлення має довжину в 256 біт (128 або менше з яких насправді будуть використовуватись).

Комбінуючи цю систему отримуємо 1024 бітове слово. Для зручності переведемо 2-чну систему числення в 16-чну систему числення та отримуємо діапазон від 00000000 до *FFFFFFFF*

Тому структура повідомлення виглядає наступним чином (рис 2.8):

Рис. 2.8. Структура основного повідомлення

Значення 2х перших цифр відповідають ознаці статусу байту та виконують функцію синхронізації.

Наступі дві цифри кодують вибрану клавішу від 15 (21 в 10чній) до 6С (108 в 10чній) (рис 2.9)

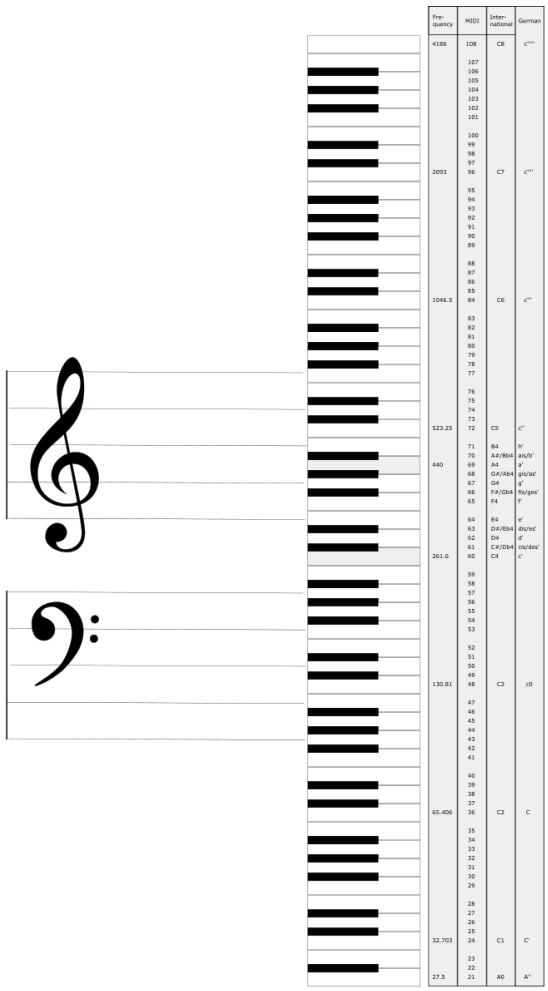

Рис. 2.9. Діапазон стандарту *MIDI*

Наступні дві цифри кодують інтенсивність натискання клавіши (якщо детектор інтенсивності існує) від 00 до *FF*. Останні дві цифри – виконують функцію службових команд.

Для того щоб визначати натискання клавіш розподілених в часі потрібен окремий потік, який змінює своє значення на більше з часом. Кожних декілька мікросекунд значення в цьому потоці буде інкрементуватись.

Для демонстрації роботи структури повідомлень використаємо програму, яка динамічно виводить на екран ці повідомлення: зліва - *MIDI*-повідомлення, справа – потік часу (рис. 2.10).

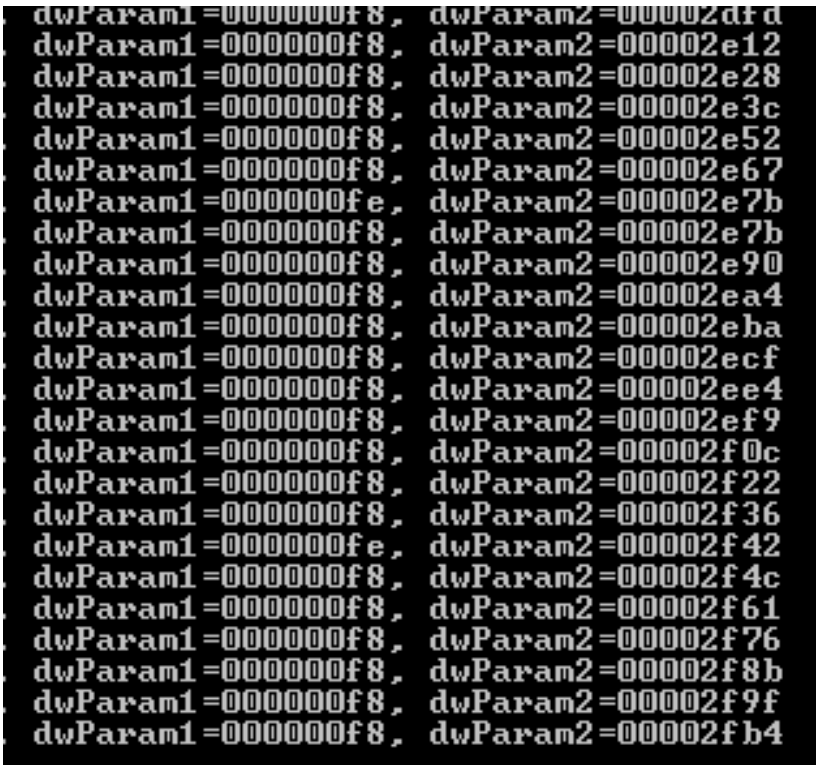

Рис. 2.10. *MIDI*-повідомлення без натискання клавіатури

Система, за якою ми будемо інтерпретувати *MIDI* сигнали. Визначає натискання та відтискання клавіш працює наступним чином: при натисканні клавіші інтенсивність натискання дорівнює ненульовому значенню. В момент відтискання клавіші *MIDI*-протокол фіксує цей момент як натиснення клавіши з нульовою інтенсивністю. (рис. 2.11)

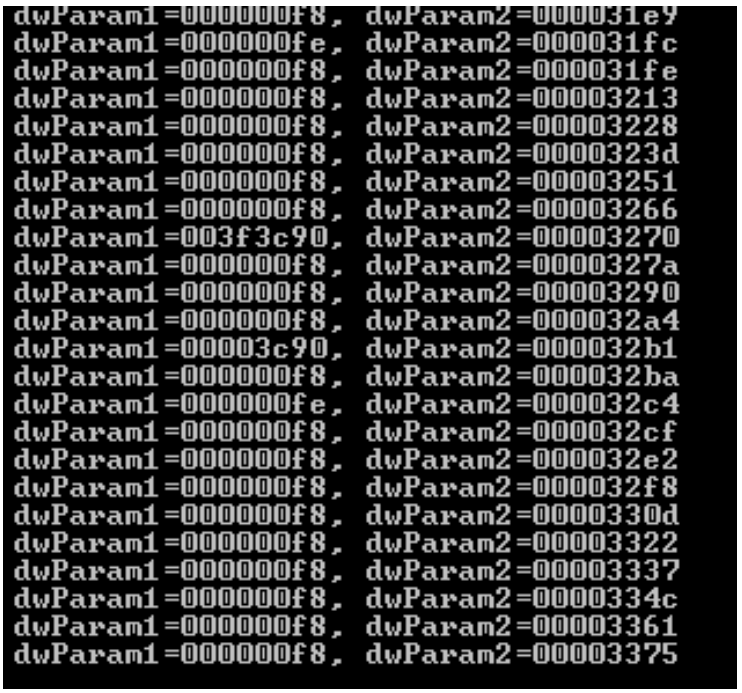

Рис. 2.11. – *MIDI* повідомлення після натискання однієї клавіши

Ненульове значення при натисканні клавіші дорівнює інтенсивності яка була визначення детектором в клавіатурі, яка інтерпретується числом від 0 до *FF* (рис. 2.12)

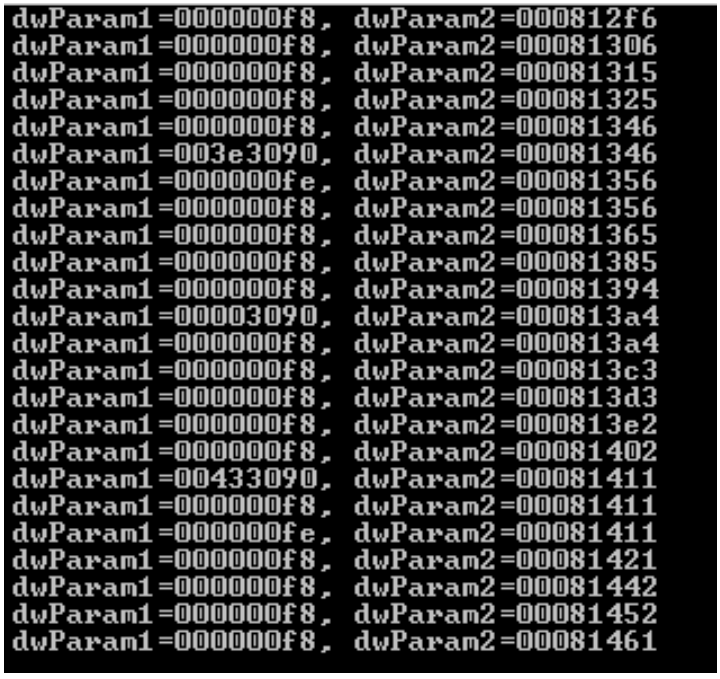

Рис. 2.12. - *MIDI* повідомлення після натискання однієї клавіші з різною інтенсивністю

MIDI-протокол може приймати одночасно декілька клавіш та порядок за якому значення клавіші зі значенням інтенсивності в часі фіксується та зникає визначає дану комбінацію та гучність ноти (рис. 2.13)

Рис. 2.13. – *MIDI* повідомлення після натискання 3х клавіш одночасно

2.3 Висновки до розділу

В другому розділі було описано та досліджено *MIDI*-протокол та зв’язані з ним технології, такі як синтезатори та сіквенсери. Показано, що *MIDI* не є форматом цифрового звуку, а послідовністю команд. Описано принцип комутації між *MIDI* приладами. Досліджено принципи та обмеження роботи комутації в технології *MIDI*. Досліджено принципи роботи *MIDI* каналів. Описано кількість та призначення *MIDI* каналів. Описана мова *MIDI* та принципи роботи повідомлень. З усіх видів повідомлень було виділено повідомлення про синхронізацію, повідомлення про вибрану клавішу та повідомлення про інтенсивність натиснення клавіші. Описана система, за якою ми будемо інтерпретувати *MIDI* повідомлення.

Також у другому розділі описано алгоритм оброки вхідних сигналів, отриманих за *MIDI* вводом для визначення тональності введених звуків.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРАКТИВНОГО ДОДАТКУ ІНТЕРПРЕТАЦІЇ ТОНАЛЬНОСТІ ТА КОМБІНАЦІЇ ТОНІВ ЗА *MIDI* ВВОДОМ

3.1.Структура інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом

Для розробки інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом модуля використовується середовище *Qt Creator*. *Qt Creator* – це інтегроване середовище розробки, призначене для створення крос-платформових використанням бібліотеки *Qt*, до якої входить велика кількість класів для вирішення тривіальних задач, а також розширені версії класів Стандартної бібліотеки шаблонів (*STL*) мови *C++*.

Підтримується розробка як класичних програм мовою *C++*, так і використання мови *QML*, для визначення сценаріїв, в якій використовується *JavaScript*, а структура і параметри елементів інтерфейсу задаються *CSS*-подібними блоками. *Qt Creator* може використовувати *GCC* або *Microsoft VC++* як компілятор і *GDB* як завантажувач. Для *Windows* версій бібліотека комплектується компілятором, заголовними і об'єктними файлами *MinGW*.

Для розробки використовуються такі модулі:

– *MIDI* (англ. *Musical Instrument Digital Interface* — протокол зв'язку між пристроєм управління, генеруючим команди, і підлеглим пристроєм, який виконує ці команди. *MIDI* дозволяє виконавцю натиснути клавішу на одному інструменті
– *Qt Core* – класи ядра бібліотеки для використання іншими модулями;
– *Qt Gui* – компоненти графічного інтерфейсу;
– *Qt Widgets* – забезпечує набір елементів інтерфейсу для створення класичних інтерфейсів у стилі настільних додатків.

Всі перераховані модулі підключаються шляхом додавання ключових слів, через пропуск, змінній *QT* у файлі з розширенням "*.pro*". Наприклад, підключення модулів *Qt Core* та *Qt Gui* виглядає наступним чином:

QT += core gui

де "*core*" та "*gui*" це ключові слова для модулів *Qt Core* та *Qt Gui* відповідно. [12, с. 342]

Наведена діаграма послідовності (рис. 3.1-3.2):

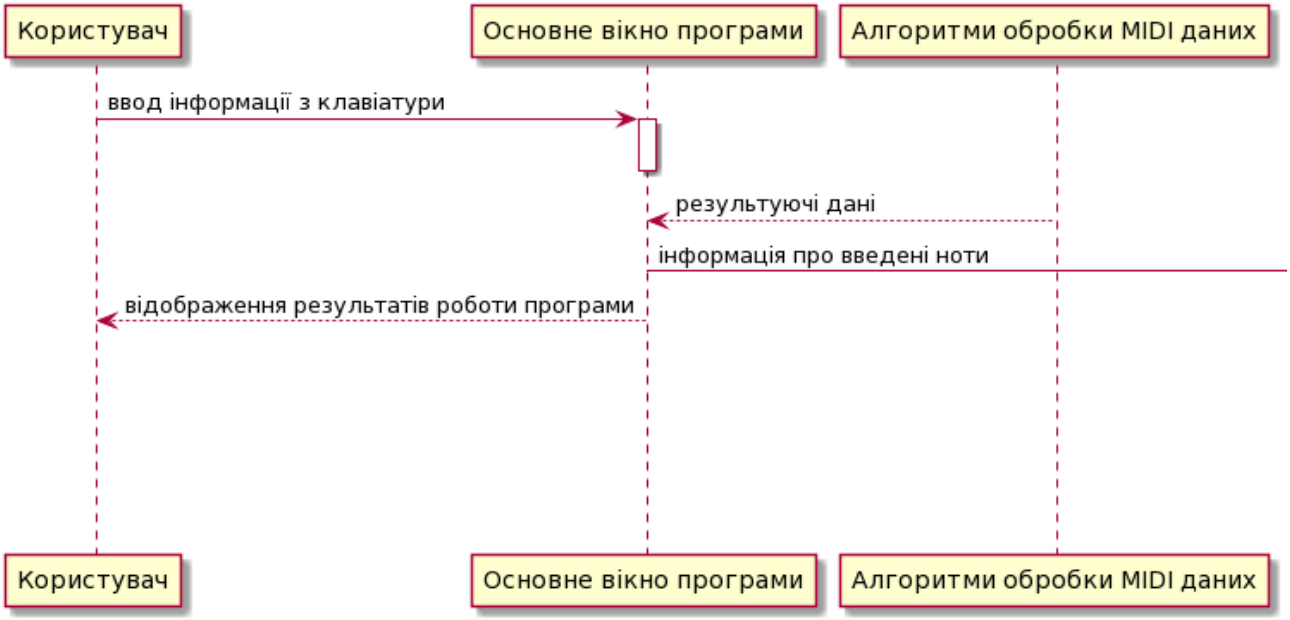

Рис. 3.1. Діаграма послідовності

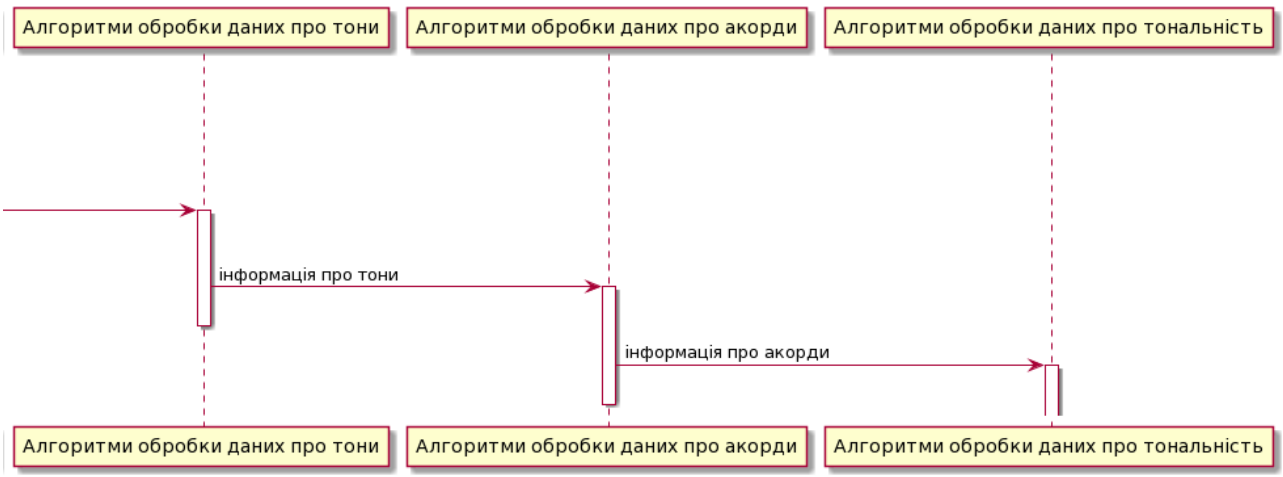

Рис. 3.2. Продовження діаграмаи послідовності

– Алгоритм обробки *MIDI*-повідомлень
– Алгоритм автоматичного пошуку поточного акорду
– Алгоритм автоматичного пошуку поточної тональності

Та декількох допоміжних алгоритмів:

– Алгоритм автоматичного пошуку інверсій (частина алгоритму пошуку поточного акорду)
– Алгоритм автоматичного пошуку акордових розширень (*extensions*) (частина алгоритму пошуку поточного акорду)
– Алгоритм визначення поточної нажатої клавіші (частина алгоритму обробки *MIDI*-повідомлень)
– Алгоритм демонстрації поточних натиснутих клавіш на екрані

Ці алгоритми передають результати роботи один одному, що поєднано функціоналом *Qt* для демонстрації результатів роботи цих алгоритмів.

Програма функціонує за такими трьома алгоритмами та декількома допоміжними алгоритмами. що програмно реалізовано у вигляді класів (рис 3.3):

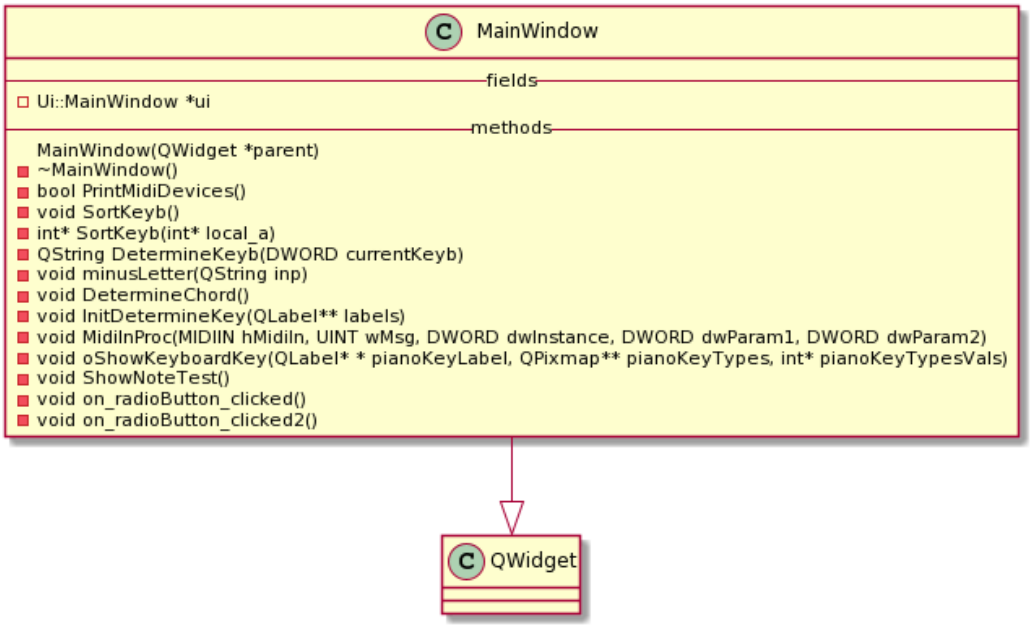

Рис. 3.3. Діаграма класів

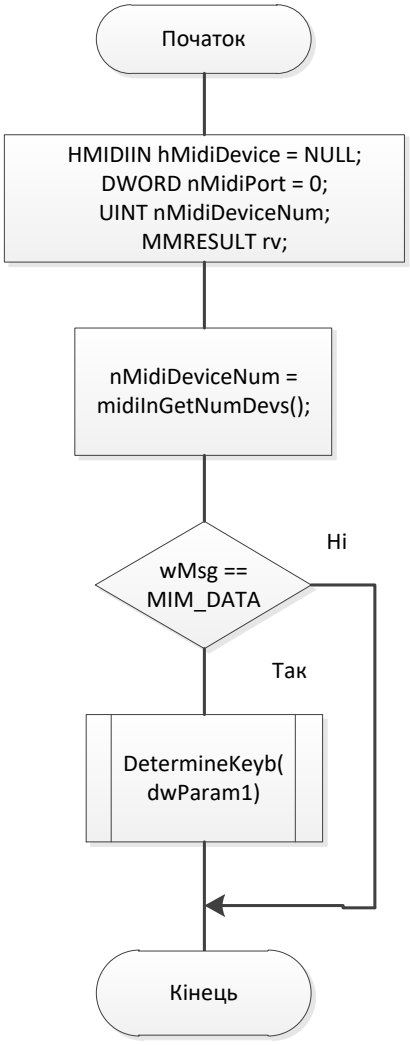

Рис. 3.4. Схема алгоритму основної частини обробки поточних *MIDI*-повідомлень

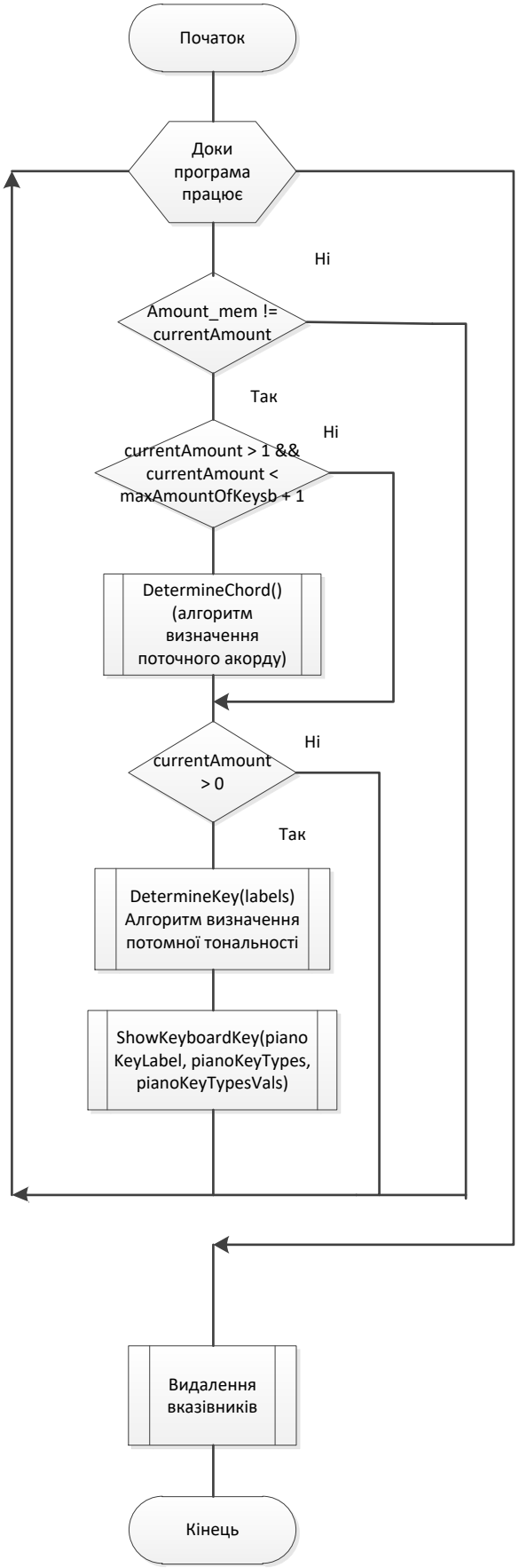

Рис. 3.5. Схема алгоритму основного циклу програми

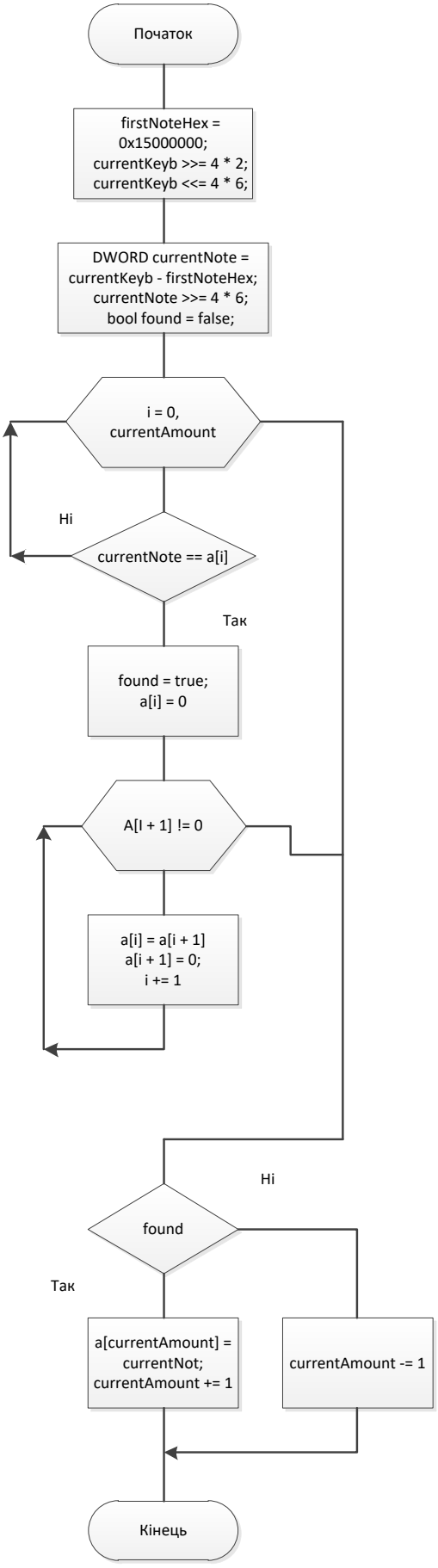

Рис. 3.6. Схема алгоритму визначення поточної нажатої клавіші

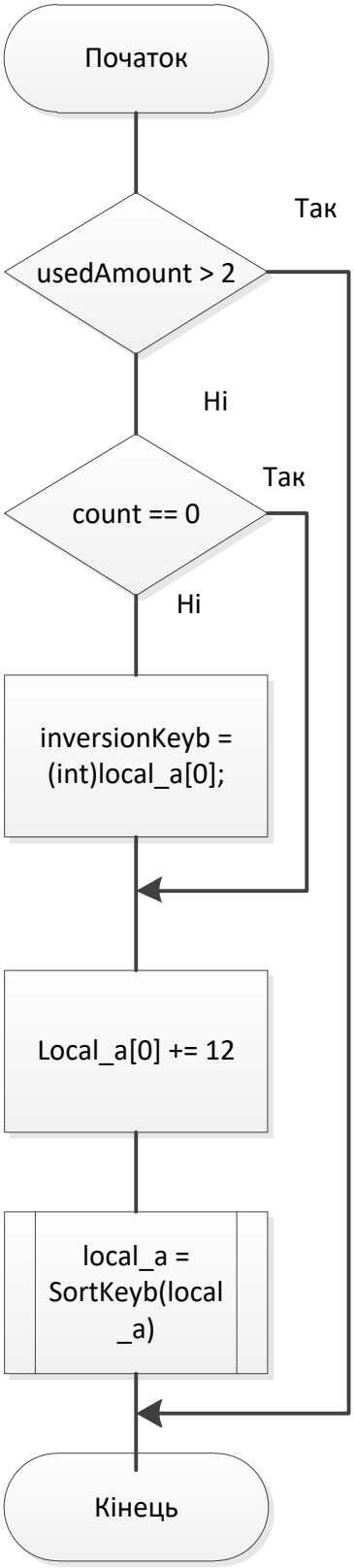

Рис. 3.7. Схема алгоритму автоматичного пошуку акордових інверсій

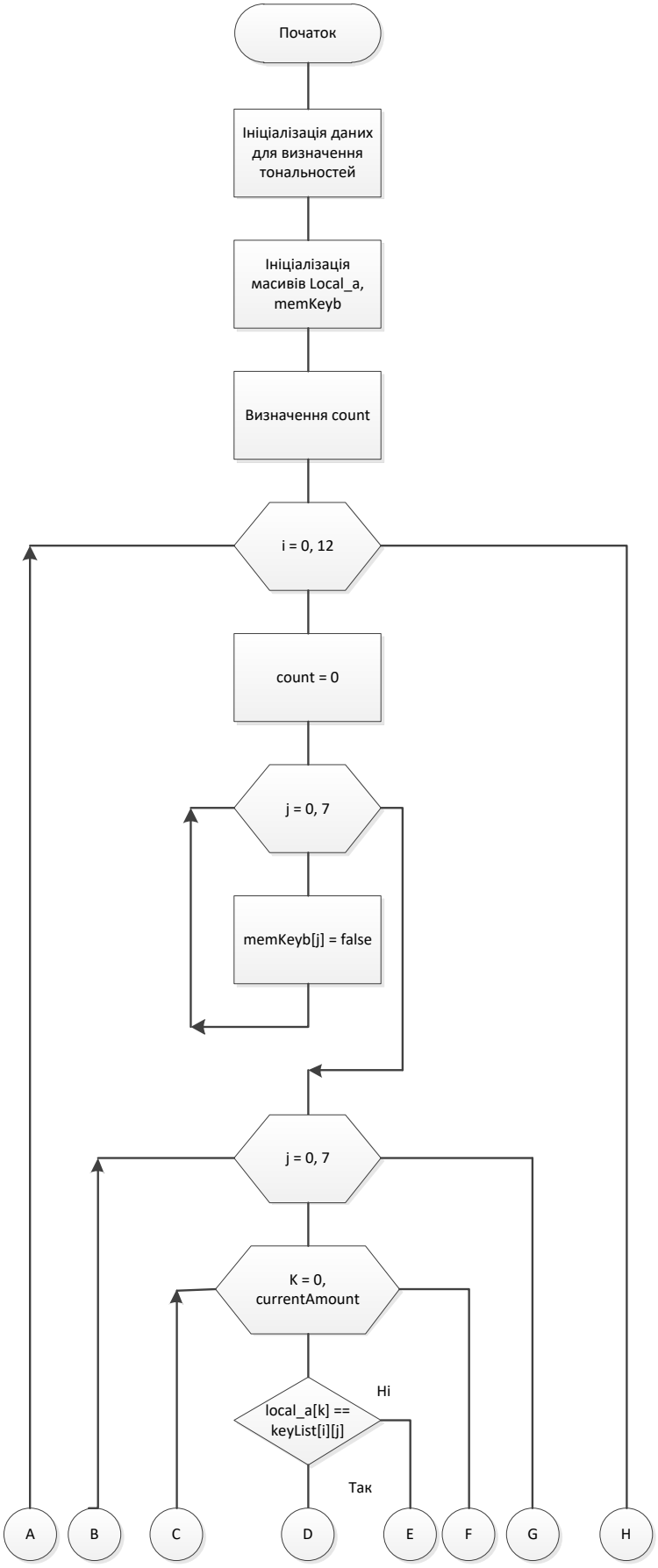

Рис. 3.8. Схема алгоритму автоматичного пошуку поточної тональності

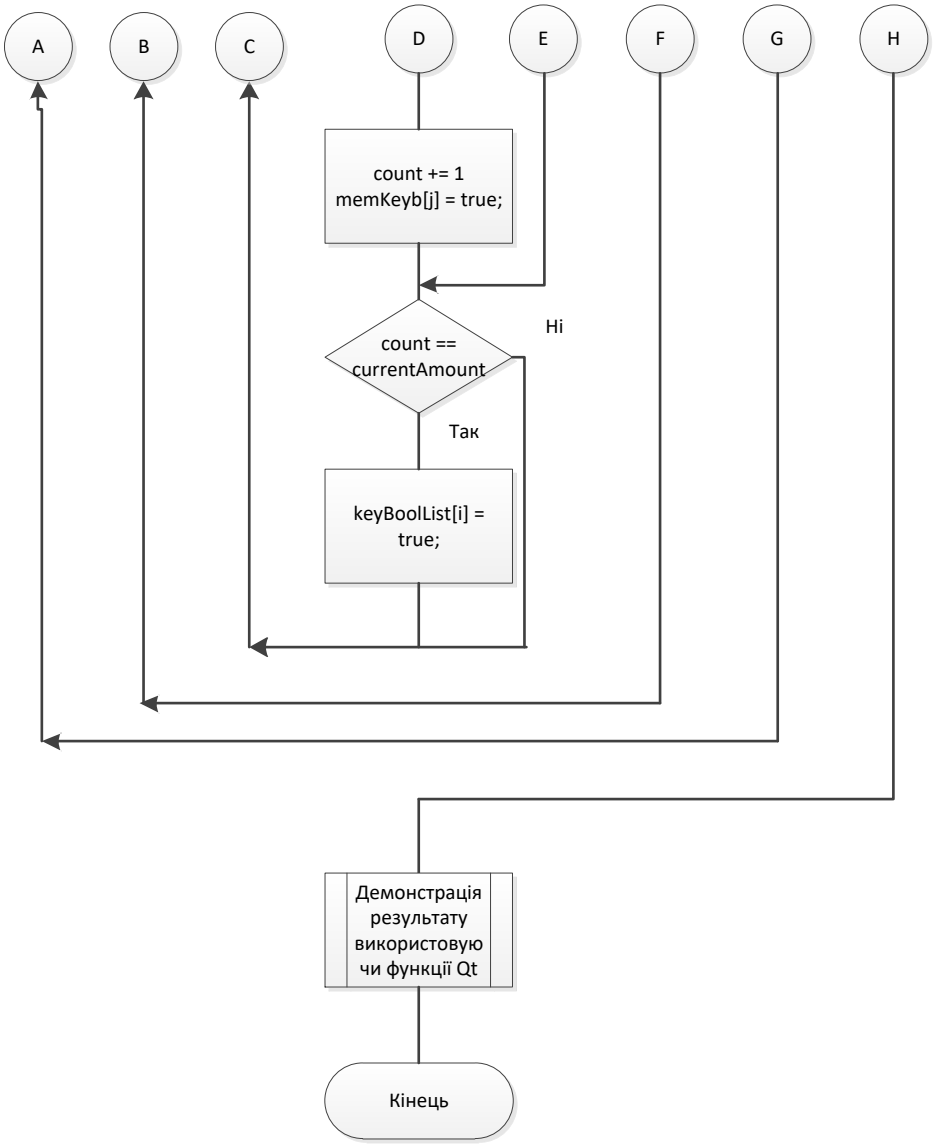

Рис. 3.9. Продовження схема алгоритму автоматичного пошуку поточної тональності

3.2. Демонстрація роботи інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом

Після підключення *MIDI* клавіатури до комп’ютеру, ми можемо починати використання функціонал програми/додатку. Інтерфейс програми зображено на рис. 3.10.

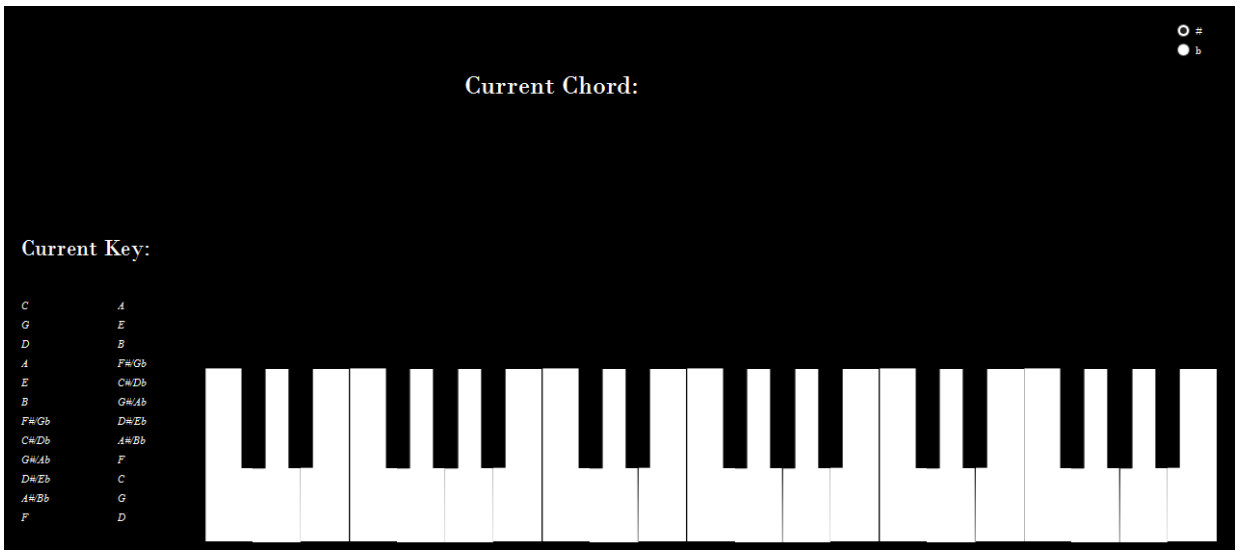

Рис. 3.10. Інтерфейс програми без введення даних

Введемо на *MIDI* клавіатурі акорд Фа мінор (ноти Фа Ля-бемоль До) (рис. 3.11).

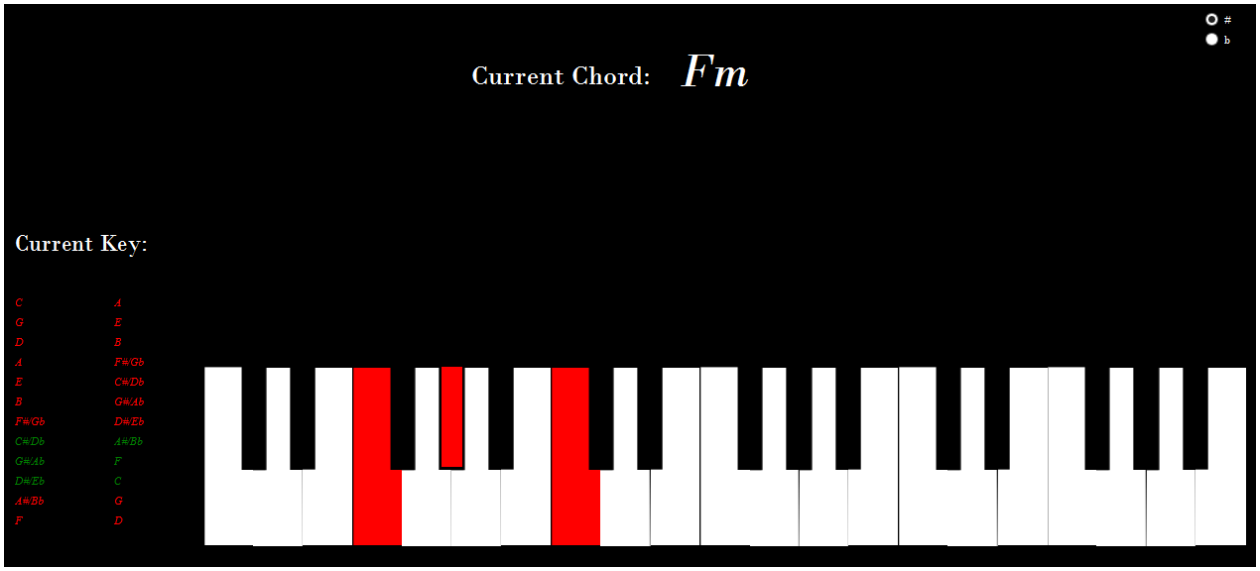

Рис. 3.11. Інтерфейс програми після введення даних з *MIDI* клавіатури

Розділ “Даний акорд” - демонструє користувачу назву акорду, який інтерактивний додаток визначає за алгоритмом розпізнавання акорду з *MIDI* клавіатури (рис. 3.12-3.13)

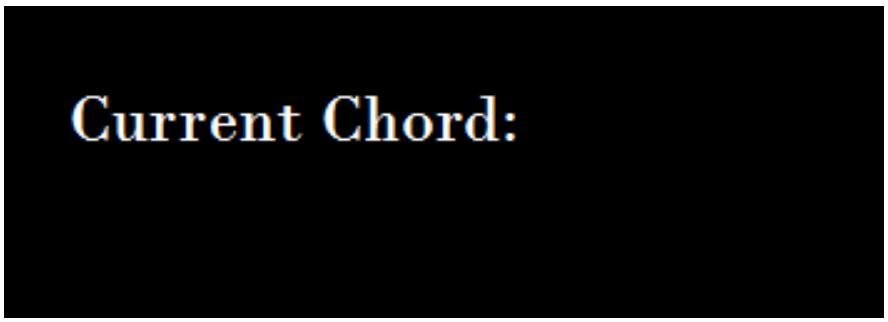

Рис. 3.12. Розділ “Клавіатура” без вводу даних

Рис. 3.13. Розділ “Клавіатура” після вводу даних

Алгоритм розпізнавання акордів визначає наступні акорди:

– Мажорний
– Мінорні
– Великий мажорний септакорд
– Малий мажорний септакорд
– Великий мінорний септакорд

Та інші комплексні акорди.

В класичній гармонії акордом вважається лише таке сполучення звуків, в якому звуки розташовані по терціях,

Нижній звук акорду називають основним тоном, інші звуки отримують назву згідно з інтервалом, що вони утворюють з основним тоном. Будь-який звук акорду може бути подвоєним (потроєним), або перенесеним в іншу октаву. Якщо основний тон акорду перестає бути нижнім — відбувається обернення акорду.

Акорд може бути розташованим тісно або широко. При тісному розташуванні сусідні голоси (крім басу) віддалені на інтервал секунди, терції,

або кварти, при широкому — на інтервал квінти, сексти або септими. Бас із тенором може утворювати будь-який інтервал. Зустрічаються також акорди в змішаному розташуванні

Приклади розпізнавання більш складних акордів (рис 3.14-3.15):

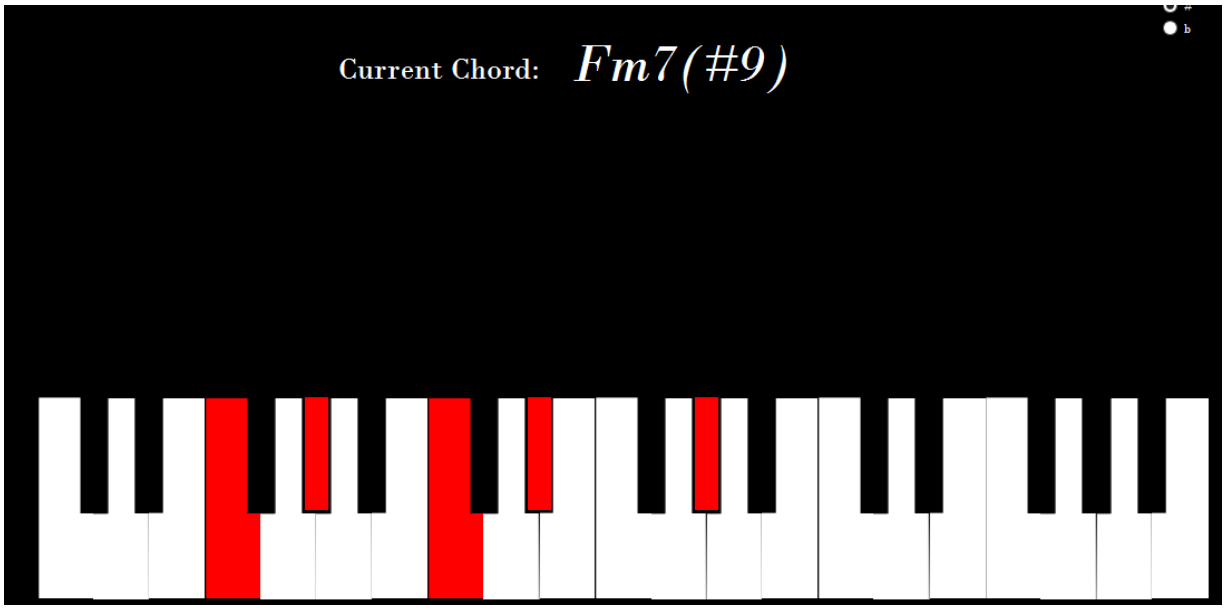

Рис. 3.14. Розпізнавання Фа малого мінорного септакорду зі збільшеним дев'ятим тоном

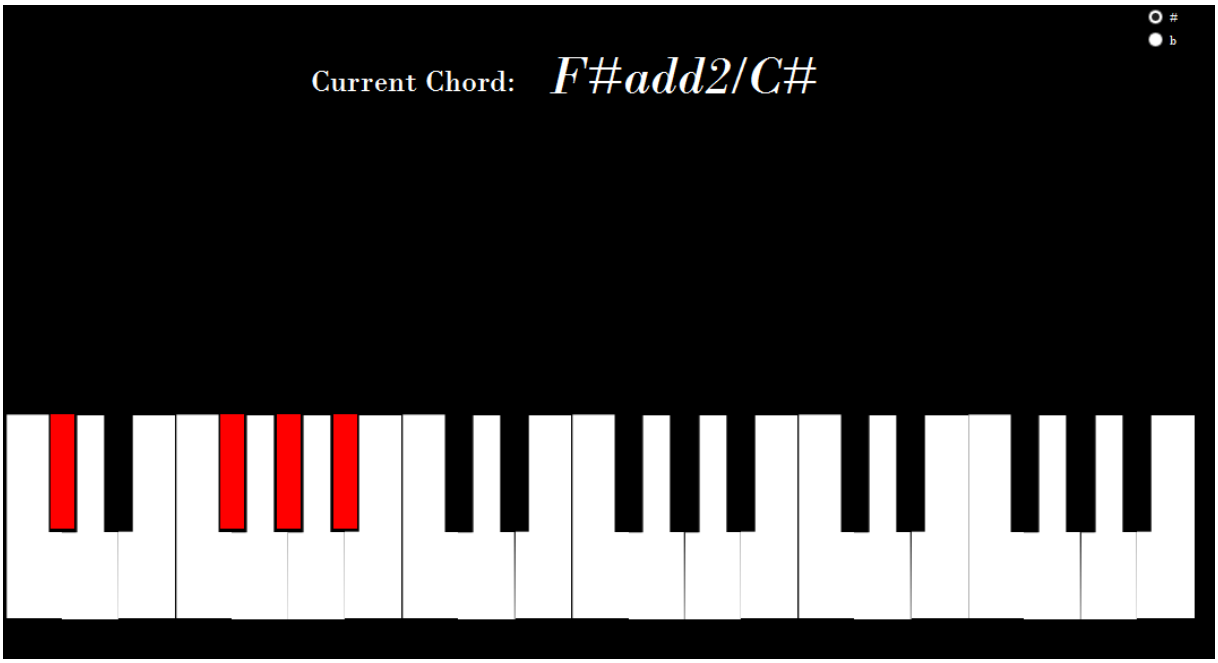

Рис. 3.15. Розпізнавання Першої інверсії Фа-діез мажору з добавленим другим тоном

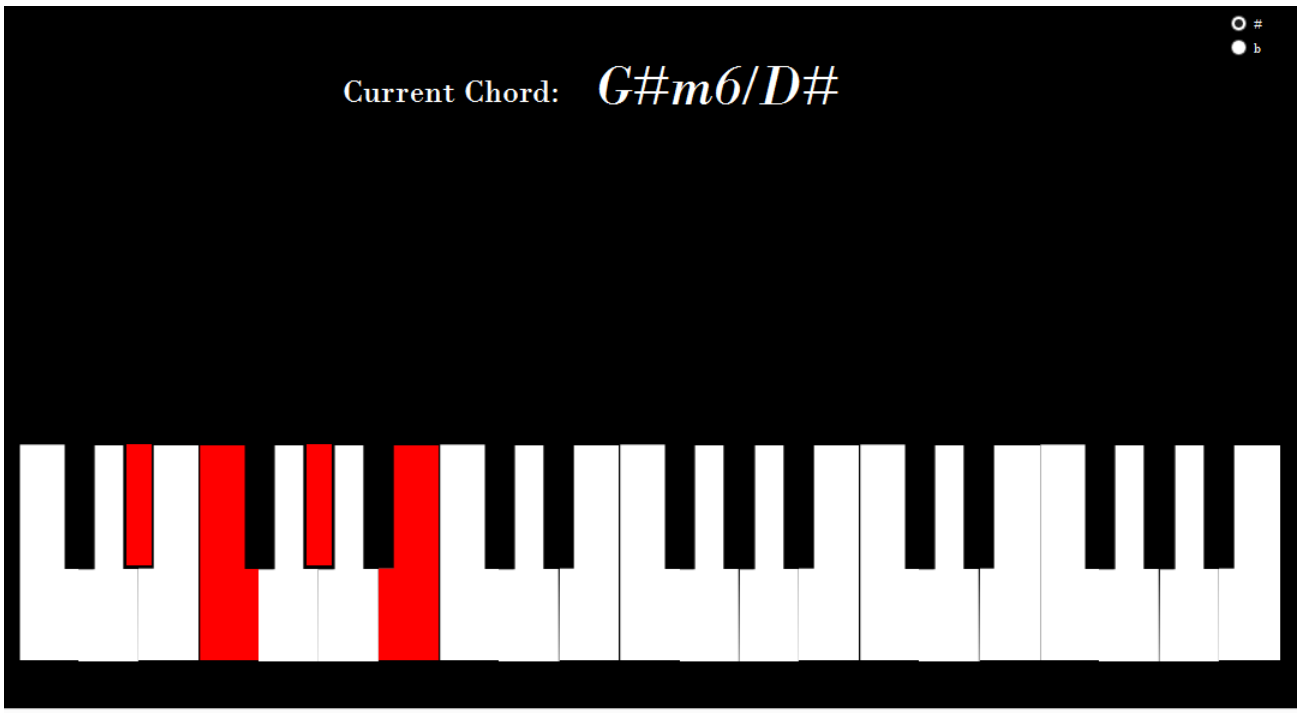

Рис. 3.16. Розпізнавання Другої інверсії Соль-діез мінорного секстакорду

Розділ “Дана тональність” - демонструє користувачу назву можливі тональності до яких дана комбінація нот/акордів підходить (рис 3.22-3.23)

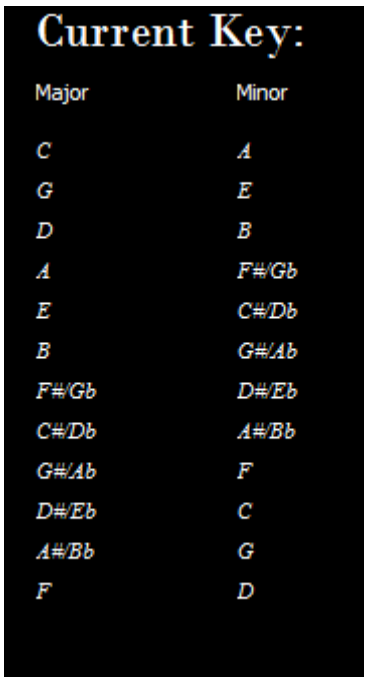

Рис. 3.17. Розділ “Тональність” без введення даних

Рис. 3.18. Розділ “Тональність” після введення акорду Фа мінор

Функціонал визначення тональності можна використовувати також не тільки для чистих акордів, а і для комбінацій нот, для перевірки “підходження” під тональність.

Наприклад, тональність Фа мінор складається з нот: Фа Соль Ля-бемоль Сі-бемоль До Ре-бемоль та Мі-бемоль. Якщо ввести всі ноти одночасно, то ми побачимо, що розділ тональність показує необхідну можливу тональність, а саме Ре-діез/Мі-бемоль мажор або Фа мінор (рис 3.24)

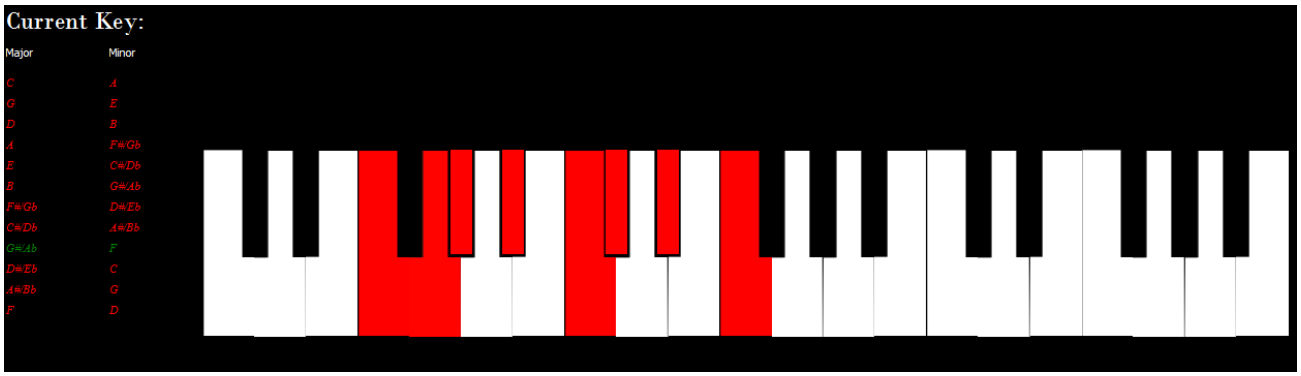

Рис. 3.19. Розділ “Тональність” після введення семи нот тональності Фа мінор

Якщо користувач точно знає тональність та потребує нотацію в бемоль виді, а не в діез. Він може переключити за допомогою кнопки справа (рис 3.25)

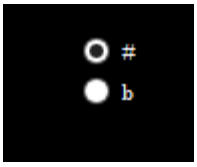

Рис. 3.20. Вигляд кнопок діез/бемоль

В залежності від вибраного режиму – отримаємо назви акордів (рис 3.26-3.27)

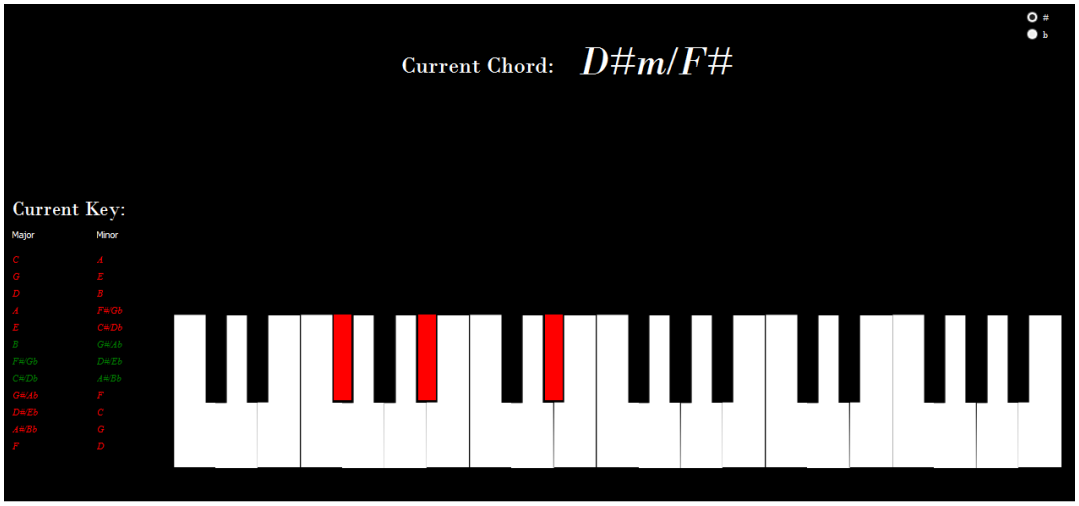

Рис. 3.21. Результат назви акорду за режимом діез

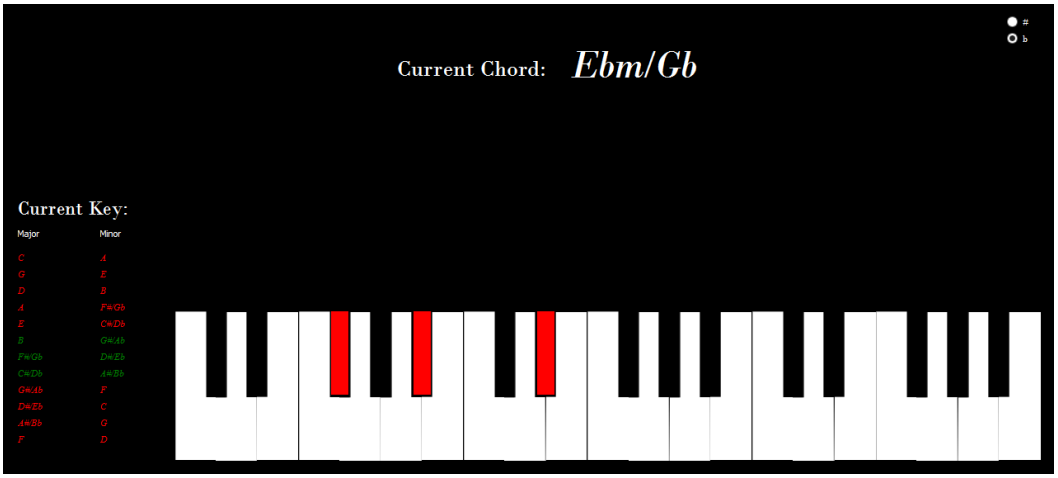

Рис. 3.22. Результат назви акорду за режимом бемоль

Протестуємо ефективність роботи програми використовуючи кількісну характеристику (за кількістю правильно визначених акордів та правильно визначеної тональності за акордом). Оберемо 30 акордів за різною складністю, структурою то тоніком.

З наступних 30 акордів (всі акорди будуть вказані за діезом): C, *Cm*. *Fm*. *Fm*7, *Gdim*, *A#maj*7, *D#mM*7, *Bdim*, *E#*, *Fmaj*7(#9), *Aaug*, *G*6(*b*9), *C*/*G*, *Baug*7, *D#m*/*F#*, *Fmaj*7/*E*, *Gm*6/*E*, *Bdim*/*F*, *Fm*6/9, *Dm*11, *Fdim*7, *D#add*4, *G#mM*7/*F*, *Eadd*4/*G*, *Fmaj*7(11), *D#m*7(*b*11), *Am*(9/11), *C*7(*b*13)

Результати:

Таблиця 3.1

Результати тесту ефективності роботи програми

	Кількість правильно визначених	В процентному співвідношенні до ідеального результату
Акордів	27 (29)	90% (96.667%)
Тональностей	30	100%

Тобто програма визначає 100% всіх тональностей або атональність для акордів.

Програма повністю не визначає 3 види акордів, 2 з них це акорди виду *dim*7 (*dim*9 *dim*11) та *aug*7 (*aug*9 *aug*11). Але визначає той факт, що вони є видами *dim* або *aug*. Тому це не є прямою помилкою роботи а програми, а лише неточність відображення результату. Третій вид акордів який програма не визначає це акорди виду з розширенням без 7мого порядку, але ті ж акорди з включенням 7мого порядку визначає безпомилково. Причина в цьому є складність виділення одного акорду від іншого якщо довжина між сусідніми нотами є далекою. Програма використовує алгоритм визначення акордів за довжиною за півтонами, тому визначення акордів типу 7(#9), 7(*b*9), 9(#11), 7(13), 11(*b*13) та подібні потребують7мий порядок (септиму).

3.3 Висновки до розділу

В третьому розділі було описано структуру інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом. Описано основні три алгоритми, а саме: алгоритм автоматичної обробки *MIDI*-повідомлень, алгоритм автоматичного пошуку поточного акорду та алгоритм автоматичного пошуку поточної тональності. Також, описані додаткові алгоритми, а саме: алгоритм автоматичного пошуку інверсій, алгоритм автоматичного пошуку акордових розширень (*extensions*), алгоритм пошуку поточних натиснутих клавіш. Було створено схеми алгоритму для цих алгоритмів. Продемонстровано процес роботи інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом. Детально описано інтерфейс додатку та її розділи.. Описано розділ “клавіатура” який демонструю поточні натиснуті клавіші. Описано розділ “даний акорд” який демонструє поточний акорд зі всіма розширення, інверсії. Описано розділ “тональність”, де додаток демонструє поточну мажорну/мінорну тональність. Також описані додаткові функції, такі як кнопки переключення між режимами результату в діез/бемоль виді.

ВИСНОВКИ

Дипломна робота присвячена темі «Інтерактивний додаток інтерпретації тональності та комбінацій тонів за *MIDI* вводом». В ході виконання роботи було описано та досліджено існуючі сервіси для музикантів та слухачів музики та аргументовано недоліки існуючих сервісів. Між сервісами для слухачів була здійснена порівняльна характеристика за такими параметрами як швидкодія, звучність та простота інтерфейсу. Були виділені деякі найкращі сервіси з них за п’яти бальною шкалою. Була здійснена порівняльна характеристика між існуючими музичними сервісами для музикантів та визначено, що вони не пропонують динамічної функції розпізнавання акордів та тональності в процесі виконання композиції музикантом, а також не надають функціоналу для визначення модальності. Також визначено, що існуючі сервіси для музикантів не пропонують функціоналу для динамічного визначення всіх можливих тональностей для акордів та для динамічного визначення комплексних акордів. В другому розділі було описано та досліджено *MIDI*-протокол та зв’язані з ним технології, такі як синтезатори та сіквенсери. Показано, що *MIDI* не є форматом цифрового звуку, а послідовністю команд. Описано принцип комутації між *MIDI* приладами. Досліджено принципи та обмеження роботи комутації в технології *MIDI*. Досліджено принципи роботи *MIDI* каналів. Описано кількість та призначення *MIDI* каналів. Описана мова *MIDI* та принципи роботи повідомлень. З усіх видів повідомлень було виділено повідомлення про синхронізацію, повідомлення про вибрану клавішу та повідомлення про інтенсивність натиснення клавіші. Описана система, за якою ми будемо інтерпретувати *MIDI* повідомлення.

Описано алгоритм оброки вхідних сигналів, отриманих за *MIDI* вводом для визначення тональності введених звуків. Описано структуру інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом. Описано основні три алгоритми, а саме: алгоритм автоматичної обробки *MIDI*-повідомлень, алгоритм автоматичного пошуку поточного акорду та алгоритм автоматичного пошуку поточної тональності. Також, описані додаткові алгоритми, а саме: алгоритм автоматичного пошуку інверсій, алгоритм автоматичного пошуку акордових розширень (*extensions*), алгоритм пошуку поточних натиснутих клавіш. Було створено схеми алгоритму для цих алгоритмів. Продемонстровано процес роботи інтерактивного додатку інтерпретації тональності та комбінацій тонів за *MIDI* вводом. Детально описано інтерфейс додатку та її розділи.. Описано розділ “клавіатура” який демонструю поточні натиснуті клавіші. Описано розділ “даний акорд” який демонструє поточний акорд зі всіма розширення, інверсії. Описано розділ “тональність”, де додаток демонструє поточну мажорну/мінорну тональність. Також описані додаткові функції, такі як кнопки переключення між режимами результату в діез/бемоль виді. Протестовано ефективність роботи програми використовуючи кількісну характеристику (за кількістю правильно визначених акордів та правильно визначеної тональності за акордом). Оберемо 30 акордів за різною складністю, структурою то тоніком. В результаті отримано такі результати: програма повністю не визначає тільки 3 види акордів, 2 з них це акорди виду *dim*7 (*dim*9 *dim*11) та *aug*7 (*aug*9 *aug*11). Але визначає той факт, що вони є видами *dim* або *aug*. Тому це не є прямою помилкою роботи а програми, а лише неточність відображення результату.. Програма використовує алгоритм визначення акордів за довжиною за півтонами, тому визначення акордів типу 7(#9), 7(*b*9), 9(#11), 7(13), 11(*b*13) та подібні потребують7мий порядок (септиму).

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойченко С. В., Іванченко О. В. Положення про дипломні роботи (проєкти) випускників Національного авіаційного університету. Київ : НАУ, 2017. 63 с.

2. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення : видання офіційне. Київ : Держстандарт України, 1995. 38 с.

3. 12 причин, почему *Spotify* лучший стримингоовый аудим сервис *URL*: https://1audioservice.com/spotify/pochemu-luchshiy (дата звернення 05.2021)

4. Распознавание музыки онлайн *URL*: https://neuronus.com/instruments/1225-raspoznavanie-muzyki-onlajn.html (дата звернення 05.2021)

5. Определение тональности для ленивых *URL*: https://www.jm.kiev.ua/book/export/html/729 (дата звернення 05.2021)

6. *MIDI* в деталях. Часть 1 – Основы *URL*: http://www.muzoborudovanie.ru/articles/midi/midi1.php (дата звернення 05.2021)

7. А. И. Волковец. Создание и обработка звука при разработке интерактивных приложений : Минск : БГУИР, 2018. 153 с.

8. Н. Ю. Глазырин. О задаче распознавания аккордов в цифровых звукозапиях. :: ИГУ, 2016. 16 с.

9. Н. С. Шевченко. Распознавание последовательности аккордов в цифровом звуке. : Санкт-Петербург: СПбГУ, 2016. 32 с.

10. *Official MIDI Specifications URL*: https://www.midi.org/specifications (дата звернення 05.2021)

11. Домашняя студия: *midi*-клавиатуры *URL*: http://www.midi.ru/doc/2.htm (дата звернення 05.2021)

12 Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк,О. В. Чеснокова, А. С. Чмыхало Программирование на языке C++ в среде *Qt Creator*: /— М. : *ALT Linux*, 2015. — 448 с. : ил. — (Библиотека *ALT Linux*).

ДОДАТОК А

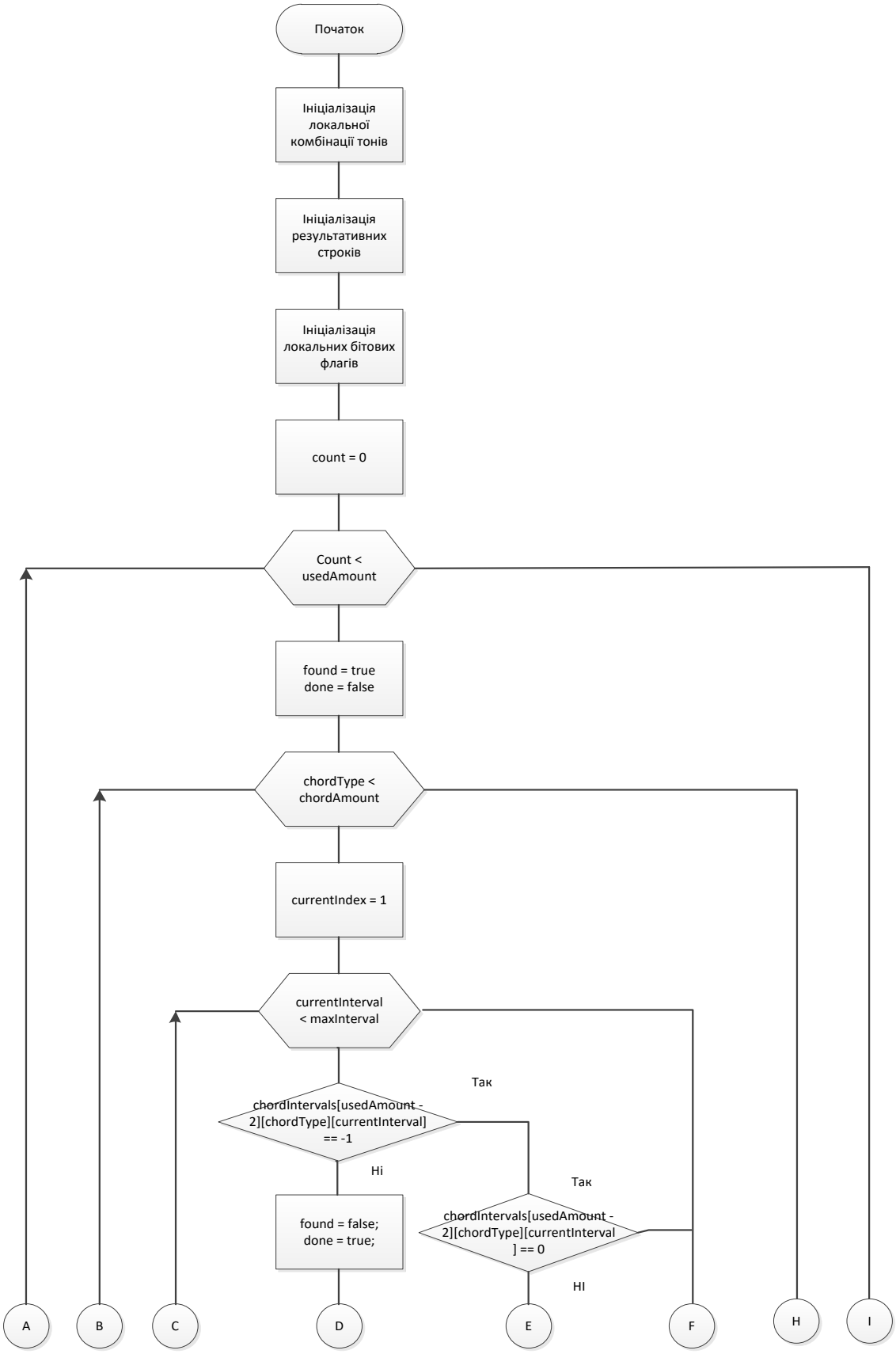

Рис. А.1 Схема алгоритму автоматичного пошуку поточного акорду

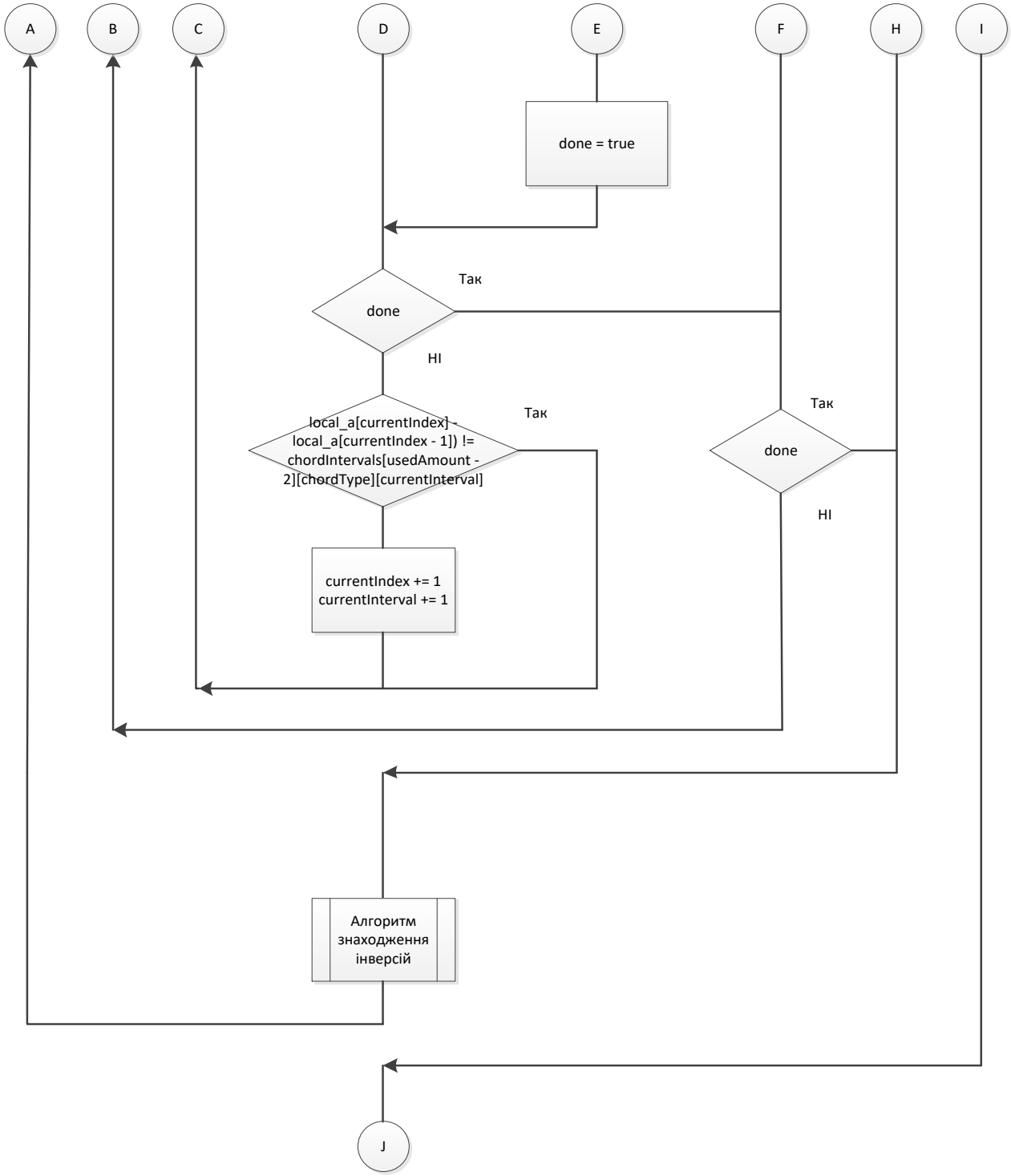

Рис А.2 Продовження схеми алгоритму автоматичного пошуку поточного акорду

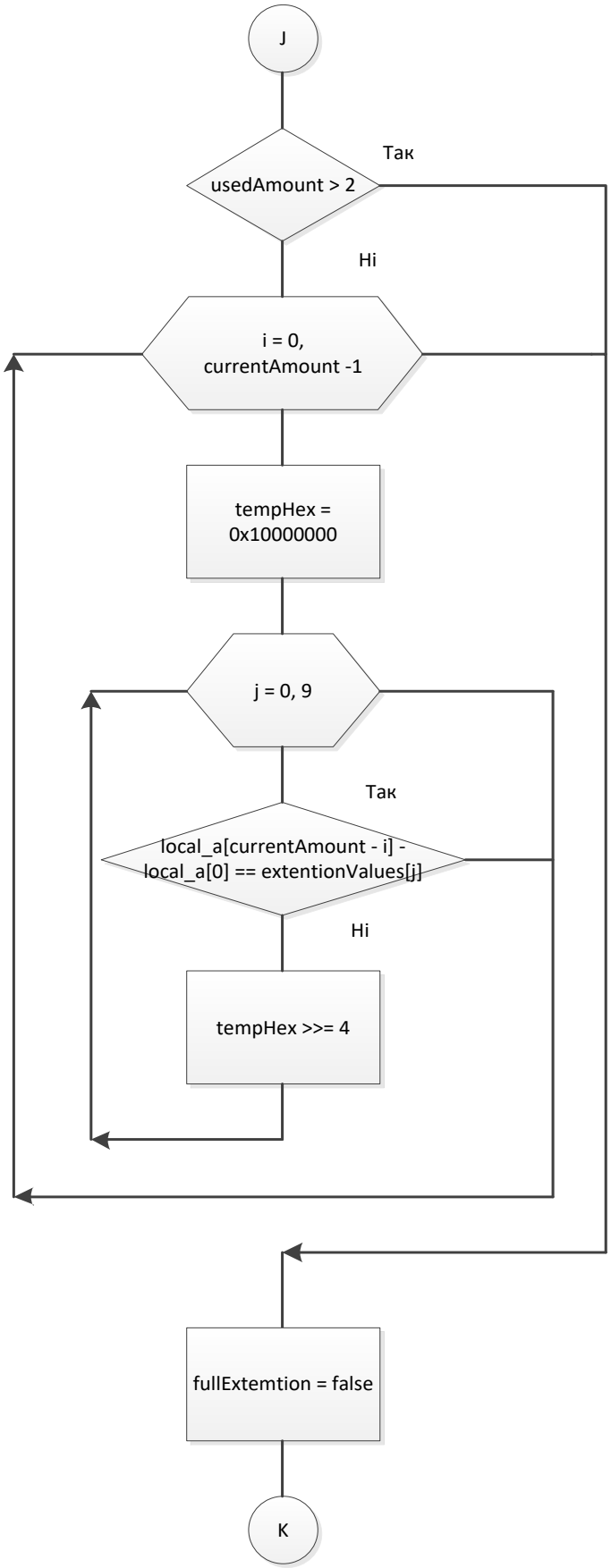

Рис. А.3. Продовження схеми алгоритму автоматичного пошуку поточного акорду

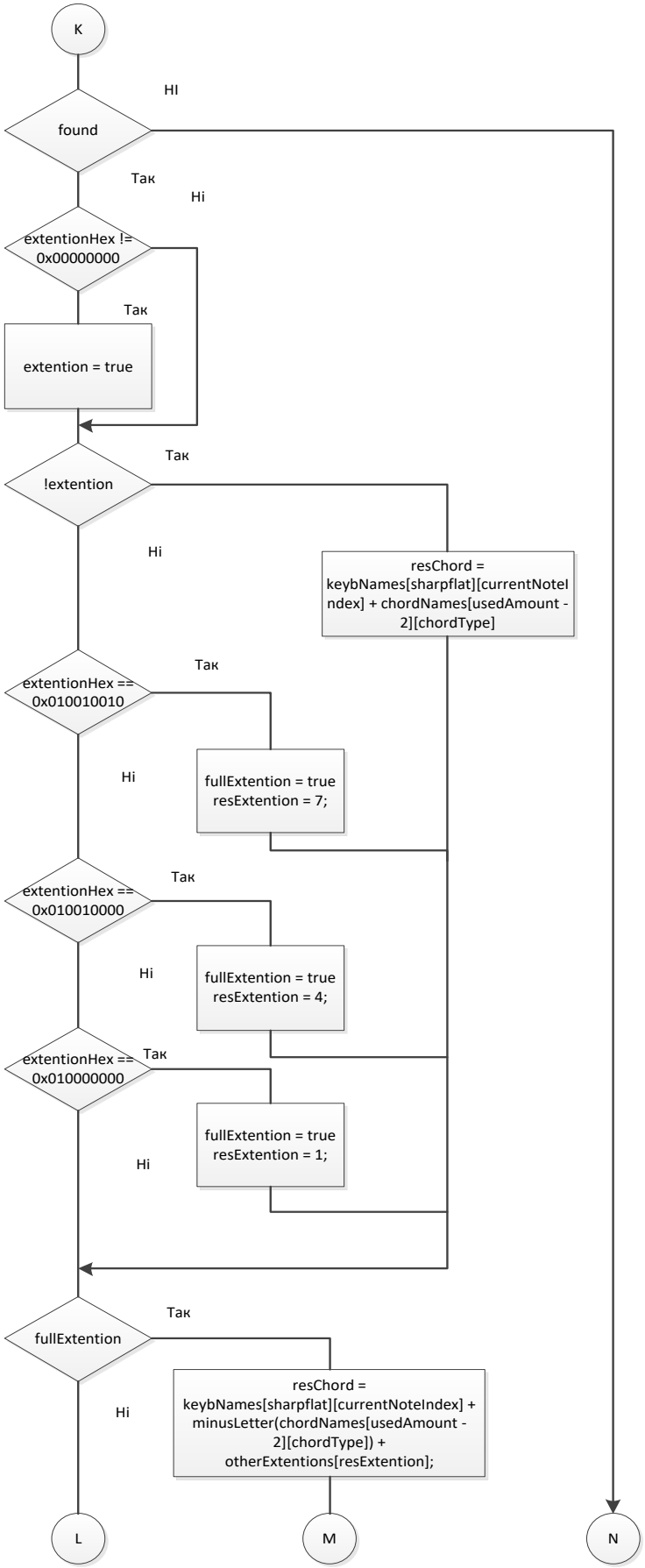

Рис. А.4. Продовження схеми алгоритму автоматичного пошуку поточного акорду

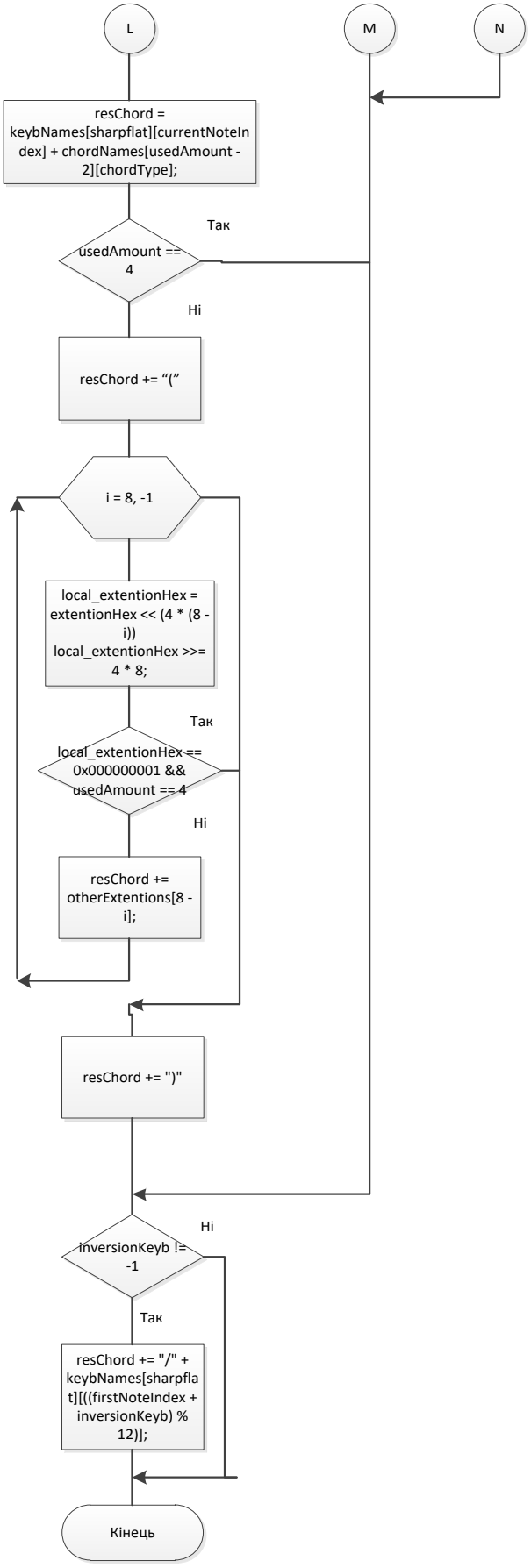

Рис. А.5. Продовження схеми алгоритму автоматичного пошуку поточного акорду

ДОДАТОК Б

```
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QFormLayout>
#include <QVBoxLayout>
#include <QLineEdit>
#include <QHBoxLayout>
#include <QRectF>
#include <QLabel>
#include <QTimer>
#include <cstdint>
#include <QApplication>
#include <QPixmap>
#include <windows.h>

const int maxAmountOfKeysb = 10;

DWORD a[15];
int currentAmount = 0;
int sharpflat = 0;
void MainWindow::DetermineChord(){
    int *local_a = new int[15];
    for(int i = 0; i < 15; i++){
        if(a[i] != 0) local_a[i] = a[i];
        else local_a[i] = 0;
        //printf("%i ",local_a[i]);
    }

    const int chordAmount = 9;
```

```
const int maxInterval = 5;

QString keybNames[2][12] = {
   {"C", "C#", "D", "D#", "E", "F", "F#", "G", "G#", "A", "A#", "B"},
   {"C", "Db", "D", "Eb", "E", "F", "Gb", "G", "Ab", "A", "Bb", "B"}
};
QString chordNames[3][chordAmount - 1] =
   {{"5", "", "", "", "", "", "", ""},
    {"", "m", "dim", "aug", "sus2", "sus4", "", ""},
    {"7", "m7", "maj7", "mM7", "6", "m6", "add2", "add4"}
};

QString otherExtentions[9] = {
   "b9", "9", "#9", "b11", "11", "#11", "b13", "13", "#13"
};
int extentionValues[9] = {
   13, 14, 15, 16, 17, 18, 20, 21, 22
};
long long int extentionHex = 0x000000000;
int chordIntervals[3][chordAmount][maxInterval] = {
   {{7, 0, 0, 0, 0}, {-1, -1, -1, -1, -1}, {-1, -1, -1, -1, -1}, {-1, -1, -1, -1, -1},
{-1, -1, -1, -1, -1}, {-1, -1, -1, -1, -1}, {-1, -1, -1, -1, -1}, {-1, -1, -1, -1, -1}, {-1, -1, -
1, -1, -1}},
   {{4, 3, 0, 0, 0}, {3, 4, 0, 0, 0}, {3, 3, 0, 0, 0}, {4, 4, 0, 0, 0}, {2, 5, 0, 0,
0}, {5, 2, 0, 0, 0}, {-1, -1, -1, -1, -1}, {-1, -1, -1, -1, -1}, {-1, -1, -1, -1, -1}},
   {{4, 3, 3, 0, 0}, {3, 4, 3, 0, 0}, {4, 3, 4, 0, 0}, {3, 4, 4, 0, 0}, {4, 3, 2, 0,
0}, {3, 4, 2, 0, 0}, {2, 2, 3, 0, 0}, {3, 2, 2, 0, 0}, {-1, -1, -1, -1, -1}}
};
int firstNoteIndex = 9;
int currentNoteIndex;
```

```
int inversionKeyb = -1;
bool found = true;
bool done = false;
int currentIndex;
int chordType;
int currentInterval;
int usedAmount;
if(currentAmount > 4) usedAmount = 4;
else usedAmount = currentAmount;
for(int count = 0; count < usedAmount; count++){
   found = true;
   done = false;
   for(chordType = 0; chordType < chordAmount; chordType++){
      currentIndex = 1;
      for(currentInterval = 0; currentInterval < maxInterval;
currentInterval++){
         if(chordIntervals[usedAmount - 2][chordType][currentInterval]
== -1){
            done = true;
            found = false;
            break;
         }
         else if(chordIntervals[usedAmount -
2][chordType][currentInterval] == 0) done = true;

         if(done) break;

         if((int)(local_a[currentIndex] - local_a[currentIndex - 1]) !=
chordIntervals[usedAmount - 2][chordType][currentInterval]){
            break;
```

```
                }
                currentIndex++;
            }
            if(done) break;
        }
        if(found) break;
        if(usedAmount > 2){
            if(count == 0) inversionKeyb = (int)local_a[0];
            //printf("%i\n", inversionKeyb);
            //printf("%i %i %i %i %i\n", local_a[0], local_a[1], local_a[2],
local_a[3], local_a[4]);
            local_a[0] += 12;

            //printf("%i %i %i %i %i\n", local_a[0], local_a[1], local_a[2],
local_a[3], local_a[4]);
            local_a = SortKeyb(local_a);
            //printf("%i %i %i %i %i\n", local_a[0], local_a[1], local_a[2],
local_a[3], local_a[4]);
        }
    }
    if(usedAmount > 2){
        long long int tempHex;
        for(int i = 0; i < currentAmount - 1; i++){
            tempHex = 0x100000000;
            for(int j = 0; j < 9; j++){
                if(local_a[currentAmount - i] - local_a[0] ==
extentionValues[j]){
                    extentionHex += tempHex;
                    break;
                }
```

```
            tempHex >>= 4;
        }
    }
}
//printf("%#09x\n", extentionHex)

QString resChord = "";
int resExtention = 0;
bool fullExtention = false;
currentNoteIndex = (firstNoteIndex + (int)local_a[0]) % 12;
bool extention = false;
if(found){
    if(extentionHex != 0x000000000) extention = true;

    if(!extention) resChord = keybNames[sharpflat][currentNoteIndex] +
chordNames[usedAmount - 2][chordType];
    else{
        switch(extentionHex){
            case 0x010010010:
                fullExtention = true;
                resExtention = 7;
                break;
            case 0x010010000:
                fullExtention = true;
                resExtention = 4;
                break;
            case 0x010000000:
                fullExtention = true;
                resExtention = 1;
                break;
```

```
                }
                if(fullExtention) resChord =
keybNames[sharpflat][currentNoteIndex] +
minusLetter(chordNames[usedAmount - 2][chordType]) +
otherExtentions[resExtention];
                else{
                    resChord = keybNames[sharpflat][currentNoteIndex] +
chordNames[usedAmount - 2][chordType];
                    if(usedAmount == 4) resChord += "(";
                    long long int local_extentionHex;
                    for(int i = 8; i > -1; i--){
                        local_extentionHex = extentionHex << (4 * (8 - i));
                        local_extentionHex >>= 4 * 8;
                        //printf("%i\n", local_extentionHex);
                        if(local_extentionHex == 0x000000001){
                            if(usedAmount == 4) resChord += otherExtentions[8 - i];
                        }
                    }
                    resChord += ")";
                }
            }
            if(inversionKeyb != -1){
              resChord += "/" + keybNames[sharpflat][((firstNoteIndex +
inversionKeyb) % 12)];
            }
            ui->label->setText(resChord);
        }
    }
    MainWindow::MainWindow(QWidget *parent)
        : QMainWindow(parent)
```

```
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    QString pianoKeyPath = "C:\\Users\\M\\Documents\\untitled2\\";
    QString pianoKeyNames[4] = {"key\\key1", "key\\key2", "key\\key3",
"key\\keyb"};
    QString pianoKeyNamesExtention[2] = {"_u.png", "_p.png"};
    QPixmap** pianoKeyTypes;
    pianoKeyTypes = new QPixmap*[4];
    for(int i = 0; i < 4; i++) pianoKeyTypes[i] = new QPixmap[2];

    for(int i = 0; i < 4; i++){
        for(int j = 0; j < 2; j++){
            pianoKeyTypes[i][j].load(pianoKeyPath + pianoKeyNames[i] +
pianoKeyNamesExtention[j]);
            if(i == 3) pianoKeyTypes[i][j] =
pianoKeyTypes[i][j].scaled(pianoKeyLabel[1]->size(), Qt::KeepAspectRatio);
            else pianoKeyTypes[i][j] =
pianoKeyTypes[i][j].scaled(pianoKeyLabel[0]->size(), Qt::KeepAspectRatio);
        }
    }
    for(int i = 0; i < 36; i++){
        pianoKeyLabel[i]->setAlignment(Qt::AlignCenter);
        pianoKeyLabel[i]->setPixmap(pianoKeyTypes[pianoKeyTypesVals[i
% 12]][0]);
    }
    for(int i = 0; i < 15; i++){
        a[i] = 0;
    }
    HMIDIIN hMidiDevice = NULL;
```

```
        DWORD nMidiPort = 0;
        UINT nMidiDeviceNum;
        MMRESULT rv;
        //printf("looooooll");
        PrintMidiDevices();
        nMidiDeviceNum = midiInGetNumDevs();
        if (nMidiDeviceNum == 0) {
           fprintf(stderr, "midiInGetNumDevs() return 0...");
        }
        //void (*MidiInProc)(HMIDIIN, UINT, DWORD, DWORD, DWORD);
        midiInOpen(&hMidiDevice, nMidiPort, (intptr_t)(MidiInProc), 0,
CALLBACK_FUNCTION);
        rv = midiInStart(hMidiDevice);
        // Ініціалізація інтерфес
        QLabel* labels[24]
        InitDetermineKey(labels); // Ініціалізація панелі тональност
        ui->label->setText("")
        int amount_mem = 0
        QTimer *timer = new QTimer(this);
        connect(timer, &QTimer::timeout, this, [=]() mutable {
           //printf("AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA");
           if(amount_mem != currentAmount){
              if(currentAmount > 1 && currentAmount < maxAmountOfKeysb +
1){
                 DetermineChord();
              }
              else{
                 ui->label->setText("");
              }
```

```
            if(currentAmount > 0){
                DetermineKey(labels);
                ShowKeyboardKey(pianoKeyLabel, pianoKeyTypes,
pianoKeyTypesVals);
            }
            else{
                for(int i = 0; i < 12; i++){
                    labels[i]->setStyleSheet("QLabel {color: white}");
                    labels[12 + i]->setStyleSheet("QLabel {color: white}");
                    pianoKeyLabel[i]-
>setPixmap(pianoKeyTypes[pianoKeyTypesVals[i]][0]);
                    pianoKeyLabel[12 + i]-
>setPixmap(pianoKeyTypes[pianoKeyTypesVals[i]][0]);
                    pianoKeyLabel[24 + i]-
>setPixmap(pianoKeyTypes[pianoKeyTypesVals[i]][0]);
                }
            }
            amount_mem = currentAmount;
          //printf("\nAAAAAAAAAAAAAA\n");
          //ShowNoteTest();
           }
        });
        timer->setInterval(10);
        timer->start();
        //midiInStop(hMidiDevice);
        //midiInClose(hMidiDevice);
        //hMidiDevice = NULL;
    }
```