Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська дипломна робота

на тему: «Розробка програмного забезпечення для навчальної системи гри на гітарі»

Виконав: студент 4 курсу
групи 2ПІ-18б
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Бодовський В.Д.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Черноволик Г.О.
(прізвище та ініціали)
Рецензент: к.п.н., доц. каф. КН Бондаренко З.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри _____
«____» _____ 2022 р.

ВНТУ – 2022

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти перший бакалаврський
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
25 березня 2022 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Бодовському Владиславу Дмитровичу

1. Тема роботи – розробка програмного забезпечення для навчальної системи гри на гітарі.

Керівник роботи: Черноволик Галина Олександрівна, к.т.н., доц. кафедри ПЗ, затверджені наказом вищого навчального закладу від 24 березня 2022 року № 66.

2. Строк подання студентом роботи 13 червня 2022 року.

3. Вихідні дані до роботи: середовище розробки Visual Studio 2019, мова розробки C#, операційна система – Windows 10, базові алгоритми для конструювання гітарних акордів.

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз та постановка задачі; аналіз даних; розробка інтерфейсу програмного забезпечення; розробка моделі навчальної системи; розробка основних алгоритмів роботи системи; розробка програмного модуля конструктора акордів; розробка програмного модуля для зворотного конструювання акордів; розробка програмного додатку

для навчальної системи гри на гітарі; тестування додатку; висновки; пе посилань; додатки.

5. Перелік графічного матеріалу: графічний інтерфейс програмного додатку; алгоритм конструювання акордів; модель роботи програмного додатку; блок-схеми алгоритмів роботи програмного додатку; тестування програмного додатку.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Черноволик Г.О., к.т.н., доцент кафедри ПЗ	25.03.22	10.06.22

7. Дата видачі завдання _____25 березня 2022 року_____

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів бакалаврської дипломної роботи	Строк виконання етапів роботи	Примітка
1	Обґрунтування вибору методу розробки та постановка задач дослідження	26.03.2022 – 02.04.2022	Вик.
2	Розробка структури та алгоритмів програмного продукту	10.04.2022 – 16.04.2022	Вик.
3	Розробка модулів звичайного та зворотного конструктору акордів	20.04.2022 – 10.05.2022	Вик.
4	Тестування програмного додатку	15.05.2022 – 31.05.2022	Вик.
5	Оформлення матеріалів до захисту БДР	01.06.2022 – 10.06.2022	Вик.

Студент _____ Бодовський В.Д.
(підпис) (прізвище та ініціали)

Керівник бакалаврської дипломної роботи _____ Черноволик Г.О.
(підпис) (прізвище та ініціали)

Анотація

Бакалаврська дипломна робота складається з 48 сторінок формату А4, на яких є 44 рисунки, 3 таблиці, список використаних джерел містить 20 найменувань.

У бакалаврській дипломній роботі проаналізовано принцип роботи конструкторів гітарних акордів.

Сформульовано мету досліджень – удосконалення та спрощення процесу навчання грі на гітарі шляхом розробки модулей звичайного та зворотнього конструкторів акордів.

Запропоновано метод вирішення питання про декілька затиснутих ладів однієї струни, та визначення серед них головного, що наближує досвід користування додатком до досвіду реального затиснення акордів. Додатково до основного методу конструювання акордів розроблено метод зворотного конструювання, що суттєво допомагає знизити поріг входження для користування програмою, та доповнює головний конструктор. За рахунок простих графічних елементів знижено навантаження на систему в цілому, що дозволяє працювати з додатком на слабших моделях апаратного забезпечення.

Розроблено програмний додаток для навчальної системи гри на гітарі, який включає у себе модуль конструювання, зворотного конструювання акорду та модуль пошуку акорду по базі.

Отримані в бакалаврській дипломній роботі результати можна використати для побудови високопродуктивних систем рендерингу.

Ключові слова: акорд, методи конструювання акордів, навчальна система.

На основі проаналізованого запропоновано впровадження власної навчальної системи.

Abstract

The bachelor's thesis consists of 48 A4 pages, which contain 44 figures, 3 tables, the list of sources used contains 20 titles.

The principle of work of guitar chord designers is analyzed in the bachelor's thesis.

The purpose of the research is formulated - to improve and simplify the process of learning to play the guitar by developing modules of conventional and reverse chord constructors.

A method of solving the question of several clamped frets of one string is proposed, and the main one is determined among them, which brings the experience of using the application closer to the experience of real chord clamping. In addition to the basic method of constructing chords, the method of inverse design has been developed, which significantly helps to reduce the entry threshold for using the program, and complements the main designer. Due to simple graphical elements, the load on the system as a whole is reduced, which allows you to work with the application on weaker hardware models.

A software application for the guitar playing system has been developed, which includes a design module, a reverse chord design module, and a chord search module based on the base.

The results obtained in the bachelor's thesis can be used to build high-performance rendering systems.

Key words: chord, methods of chord construction, educational system.

Based on the analyzed, the introduction of its own educational system is proposed.

ЗМІСТ

ВСТУП

Мистецтво та музика – це невід`ємна частина людського життя. Вони можуть слугувати самовираженням людини, її відпочинком, хобі, способом вираження почуттів та емоцій, тощо. З розвитком людської культури та технологій прослуховування музики стало доступне кожному. Так само, як її самостійне відтворення за допомогою музичних інструментів.

Гітара – неофіційно найпопулярніший музичний інструмент у світі. Існує 4 види програмних додатків, що допомагають навчатися грі на ній або у її подальшому використанні:

– для налаштування струн гітари (тюнер);
– для тренувань із дотримання темпу при грі (метроном);
– для конструювання гітарних акордів;
– експериментальні додатки з синхронної гри.

Програмні додатки для налаштування струн гітари є чудовою заміною їх фізичним аналогам, тюнерам. Вони можуть бути встановлені не тільки на стаціонарні комп`ютерні системи чи ноутбуки, але й на телефон, що підвищує зручність їх використання.

Програмні додатки для тренувань із дотримання темпу при грі на музичному інструменті також прийшли на заміну їх фізичним аналогам, метрономам і також зручніші у використанні. Також можуть бути встановлені на мобільний телефон.

Системи для конструювання гітарних акордів передбачають інтерактивний графічний гриф гітари, за допомогою якого можна дізнатися назву акорду з конкретною аплікатурою. Корисні при композиторстві власних мелодій та їх нотуванні.

Додатки з синхронної гри на гітарі скоріше поки експеримент, ніж повністю працюючі системи. Вони передбачають відтворення людиною у реальному часі мелодії, яка показана на екрані пристрою та прослуховування її системою за

допомогою мікрофону. Але, труднощі, пов`язані з помилками подібних програмних додаткві через специфіку мікрофонів, акустики приміщення та затримкою обробки інформації, ще не вирішені до кінця.

Найдоцільнішим для навчання буде використовувати саме конструктори гітарних акордів. Основним їх недоліком є те, що вони призначені для більш досвідчених користувачів та музикантів та погано підходять для новачків. Для розі`вирішення цієї проблеми доцільною є розробка додаткового модулю зворотнього конструктора акордів. Тому актуальною є розробка навчальної системи гри на гітарі, яка буде містити у собі обидва модулі.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою бакалаврської дипломної роботи є удосконалення та спрощення процесу навчання грі на гітарі шляхом розробки модулей звичайного та зворотнього конструкторів акордів.

Основними задачами роботи є:

– розробити метод та алгоритм роботи інтерактивного грифу гітари;
– розробити алгоритм роботи конструктора акордів;
– розробити алгоритм роботи зворотнього конструктора акордів;
– розробити зручний та зрозумілий графічний інтерфейс;
– розробити програмний додаток для навчальної системи гри на гітарі;
– провести тестування додатку згідно поставлених задач.

Об’єкт дослідження – процес розробки програмного додатку.

Предмет дослідження – програмний продукт для навчання грі на гітарі.

Методи дослідження. У процесі досліджень використовувались методи дослідження:

– методи створення електронних навчальних систем;
– методи передачі, кодування та обробки комп`ютером інформації;

– методи автоматизації конструювання та зворотнього конструювання гітарних акордів;
– юзер-френдлі методи подання інформації.

Наукова новизна отриманих результатів. Подальшого розвитку отримав метод конструювання гітарних акордів, який відрізняється наявністю додаткового метода зворотного конструювання акордів та, навідміну від існуючих аналогів, дозволяє працювати без інтернет-підключення та використання сторонніх баз даних.

Практична цінність отриманих результатів. Практична цінність полягає у кінцевій реалізації програмного додатку із вбудованими інструментами конструктору та зворотнього конструктору акордів та може використовуватися музикантами початківцями.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській дипломній роботі, отримані автором особисто. У науковій роботі, опублікованій у співавторстві, автору належать такі результати: модуль для конструювання акордів; модуль зворотнього конструювання акордів; алгоритми роботи інтерактивного грифу.

Аналіз. У пояснювальній записці до бакалаврської дипломної роботи було розглянуто 4 розділи та було використане ? літературне джерело.

1 ОБҐРУНТУВАННЯ ВИБОРУ МЕТОДУ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз стану навчальних систем гри на гітарі

Незалежно від обставин та часу, творчість залишається важливою частиною людського життя. Завжди будуть з`являтися нові люди, що хотітимуть опанувати гру на гітарі. Сучасні автоматизовані системи навчання дають наступні можливості:

– самостійне налаштування інструменту;
– правильна та своєчасна подача теоретичного матеріалу;
– використання інструментів, що допомагають при навчанні [1].

Через значний розвиток технологій стало можливим навчання не тільки завдяки інформації, отриманій у книгах, або напряму від іншої людини, але й за допомогою використання різних електронних навчальних систем. Через швидкий темп життя та нестачу вільного часу люди частіше й частіше вдаються саме до такого методу навчання. Адже електронні навчальні системи доступні кожному будь-де та будь-коли.

Є декілька головних рис хорошої електронної навчальної системи:

– зрозумілий та зручний графічний інтерфейс;
– правильне дозоване подання інформації;
– широкий функціонал, який не зіб`є користувача з пантелику.

Оскільки більшість подібних систем навчання грі на гітарі не роблять наголосу на конструкторі або зворотньому конструкторі акордів, є актуальною розобка власного програмного додатку, що спеціалізуватиметься на цьому.

1.2 Порівняльний аналіз аналогів

Існує декілька аналогів конструктору акордів:

– Chords.top;
– Mychords.net;

– U-rock.com.ua.

Chords.top (рис. 1.1) – веб-система з інтерактивним грифом гітари, що дозволяє дізнатися найменування акорду. Особливості цього генератора в тому, що він працює онлайн, миттєво визначаючи акорд по заданій аплікатурі і не вимагає додаткового натискання інших кнопок. Іншою особливістю є його адаптивність, адже гриф онлайн-генератора розташований вертикально, за рахунок чого забезпечується зручність використання цього інструменту на мобільних пристроях. Недоліком цього конструктору гітарних акордів є ту, що він повністю залежний від інтернет підключення, що знижує зручність його використання та доступність у регіонах зі слабким інтернетом, або його відсутністю [2].

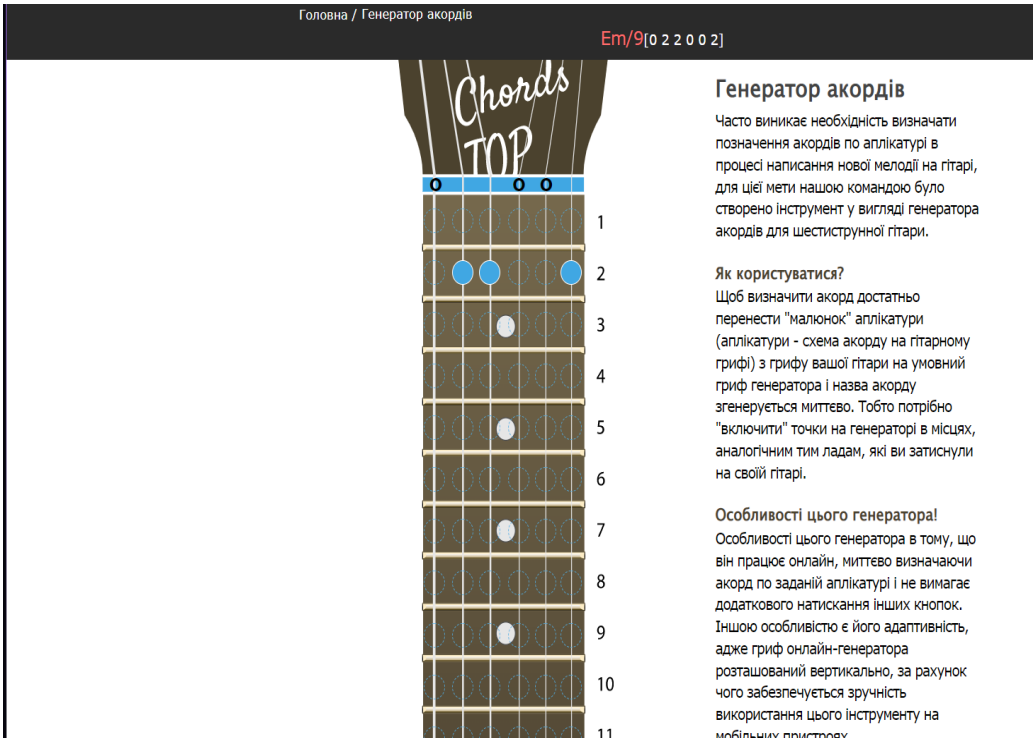

Рисунок 1.1 – Головна сторінка конструктору Chords.top

Mychords.net (рис. 1.2) – веб-система, основним функціоналом якої є запис до власної бази даних тексту та акордів нових пісень, а також їх збереження. Також система містать свій зворотній конструктор акордів, що має вигляд декількох випадаючих списків та картинок з аплікатурою. У системі повністю відсутній конструктор акордів, що значно збільшує час пошуку потрібної

аплікатури для досвідчених користувачів і робить його неможливим для новачків. Також є залежним від підключення до мережі інтернет [3].

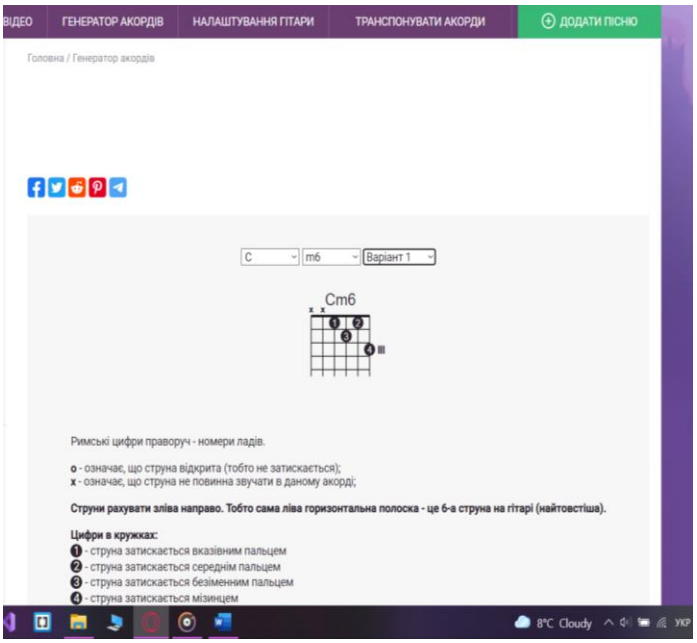

Рисунок 1.2 – Сторінка «Генератор акордів» сайту Mychords.net

U-rock.com.ua (рис. 1.3) – конструктор цієї веб-системи дуже схожий з попереднім. Ця веб-система більш направлена на навчання, оскільки містить у собі допоміжні інструменти: метроном, камертон та тюнер. Також система пропонує платні курси гри на гітарі. Залежна від системи інтернет [4].

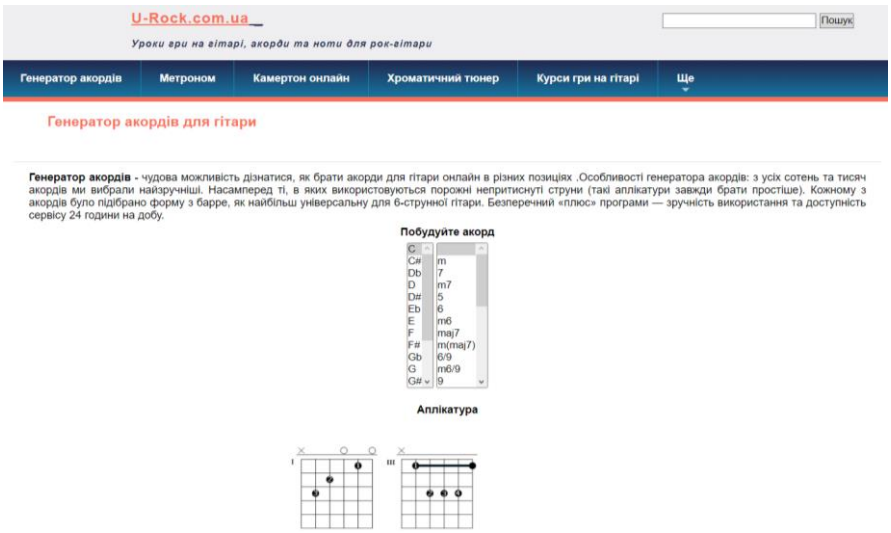

Рисунок 1.3 – Сторінка «Генератор акордів» сайту U-rock.com.u

Результати порівняння аналогів зведено в табл. 1.1.

Таблиця 1.1 – Порівняльні характеристика систем

Критерії	Chords.top	Mychords.net	U-rock.com.ua	Власний модуль
Наявність конструктору акордів	+	-	-	+
Наявність зворотнього конструктору акордів	-	+	+	+
Адаптивний інтерфейс	+	+	+	-
Наявність інтерактивного грифу	+	-	-	+
Незалежність від інтернет підключення	-	-	-	+
Загальна оцінка	60%	40%	40%	80%

Згідно з таблицею порівняльних характеристик розробка власного програмного забезпечення для навчальної системи є доцільним та актуальним. Кінцевий продукт зможе конкурувати з аналогами та виконуватиме функції звичайного та зворотнього конструктору гітарних акордів.

1.3 Аналіз методів розв’язання поставленої задачі

Найшвидшим методом вирішення поставленої задачі буде додати до одної з уже існуючих систем частин, яких їй не вистачає. Але у такому разі більша частина роботи покладається саме на розробників існуючої системи. Тому краще обрати інший варіант [5].

Альтернативним методом є розробка власного програмного додатку для інтеграції потрібних модулів. Такий підхід дає можливість приділити більше уваги конкретним блокам або модулям додатку. Ще одна перевага такого підходу – повна автономність та незалежність від інших існуючих систем, що підвищує надійність програмного продукту [6].

У випадку розробки повноцінної власної системи із влаштованим покращеним модулем зникає необхідність адаптувати додаток під різні потреби сторонніх кільки потребує повноцінної розробки системи, а тому є трудомістким. Розробка програмного забезпечення має справу з проблемами якості, вартості та надійності. Деякі програми містять мільйони рядків вихідного коду, які, як очікується, повинні правильно виконуватися в умовах, що змінюються. Складність ПЗ порівнянна зі складністю найбільш складних з сучасних машин, таких як літаки [7].

Рисунок 1.1 – Microsoft Azure Cognitive Services

Врахувавши переваги та недоліки кожного методу було вирішено скористатися методом розробки власного програмного додатку із влаштованим модулем звичайного та зворотнього конструктору акордів. Це дозволить створити повноцінний програмний продукт без необхідності використання потужного апаратного забезпеченя.

1.4 Аналіз задач для програмного додатку

Проаналізувавши переваги та недоліки існуючих веб-систем, було сформульовано такі задачі:

- розробити метод та алгоритм роботи інтерактивного грифу гітари;
- розробити алгоритм роботи конструктора акордів;

- розробити алгоритм роботи зворотнього конструктора акордів;
- розробити зручний та зрозумілий графічний інтерфейс;
- розробити програмний додаток для навчальної системи гри на гітарі;
- провести тестування додатку згідно поставлених задач.

1.5 Висновки

У першому розділі було розглянуто актуальний стан системи для навчання грі на гітарі. Було порівняно навчальну систему з відомими аналогами, такими як: Chords.top, Mychords.net та U-rock.com.ua. У результаті аналізу переваг та недоліків існуючих систем було виявлено, що усі вони залежні від інтернет підключення, а також не містять у собі усього потрібного функціоналу одразу. На основі цього було прийнто рішення розробити власний програмний додаток. За допомогою отриманої інформації було створено перелік задач, що необхідно виконати для розробки власного програмного додатку.

2 РОЗРОБКА СТРУКТУРИ ТА АЛГОРИТМІВ ПРОГРАМНОГО ПРОДУКТУ

2.1 Аналіз даних

Кожний програмний додаток чином працює з даними. Додаток приймає на вхід дані у форматі, зрозумілому для людини, кодує або компілює їх, опрацьовує, після чого виводить результат [8].

Стандарт кодування – це схема нумерації, згідно з якою кожному текстовому символу в наборі відповідає певне числове значення. Набір символів може містити буквені, числові та інші символи. Зазвичай мови складаються з різних наборів символів, тому для їх відображення передбачено цілу низку стандартів кодування [9].

ASCII стандарт кодування тексту використовує 128 унікальних значень (0–127) для представлення алфавітних, числових та пунктуаційних символів, які зазвичай використовуються в англійській, плюс вибір елементів які не представляють символів для друку. Наприклад, велика літера A має ASCII-символ 65, цифра 2 - ASCII 50, символ } має ASCII 125. Системи на базі ASCII використовують сім бітів для цифрового представлення цих значень [10].

На відміну від них, більшість комп’ютерів зберігають дані в пам’яті, організованій у восьмирозрядних байтах. Файли, що містять машинно-виконуваний код та нетекстові дані, зазвичай містять усі 256 можливих восьмибітових значень байтів. Багато комп'ютерних програм почали покладатися на цю різницю між семирозрядними і восьмирозряднми і не функціонуватимуть належним чином, якщо символи, що не належать до ASCII, з’являються в даних, які, як очікується, включатимуть лише текст ASCII. Наприклад, якщо значення восьмого біта не зберігається, програма може інтерпретувати значення байту вище 127 як прапор, що повідомляє йому про виконання якоїсь функції [11].

Однак часто бажано мати можливість надсилати нетекстові дані через текстові системи, наприклад, коли можна додати до файлу зображення у

повідомленні. Для цього дані кодуються таким чином, що восьмирозрядні дані кодуються в семирозрядні символи ASCII (зазвичай використовують лише буквено-цифрові та пунктуаційні символи). Після безпечного прибуття до місця призначення вони декодуються назад у свою восьмирозрядну форму. Цей процес називають двійковим кодуванням тексту. Багато програм виконують це перетворення, щоб забезпечити перенесення даних, наприклад PGP і GNU Privacy Guard (GPG).

Щоб уникнути проблем із кодуванням і розшифруванням тексту, гарним варіантом буде використання кодування його Юнікодом. Цей стандарт кодування містить більшість наборів символів з усіх мов, які зазвичай використовуються на сучасних комп'ютерах. Наприклад, Microsoft Word працює на основі Юнікоду, тому автоматично зберігає файли саме з цим кодуванням. Файли в Юнікоді можна відкривати й читати на персональному комп`ютері з українською мовою операційної системи незалежно від мови тексту. Крім того, є можливим збереження на такому комп'ютері файли з кодуванням Юнікод, що містять символи, яких немає в кириличних абетках, як грецькі, арабські, японські чи західноєвропейські [12].

Кращим варіантом для використання при розробці програмного додатку для навчальної системи гри на гітарі буде схема нумерації, що реалізована у полях введення тексту для його кодування та обробки.

2.2 Розробка структури інтерфейсу програмного додатку

При розробці графічного інтерфейсу користувача було використано невелику кількість елементів керування [13].

Інтерфейс додатку умовно поділений на декілька частин. У верхній частині вікна програми знаходиться інтерактивний гітарний гриф (рисунок 2.1), що слугує частиною конструктору гітарних акордів [14]. Нижче знаходяться три кнопки та чотири текстових поля, які також відповідають за роботу конструктора, а також зворотнього конструктора.

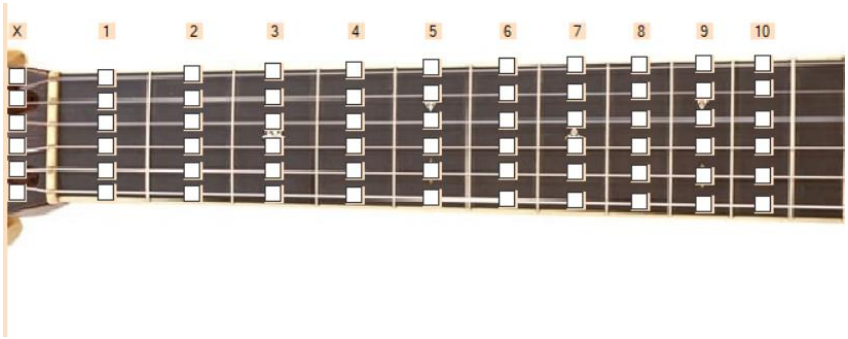

Рисунок 2.1 – Інтерактивний гриф гітари

Одна із кнопок відповідає за початок процесу перетворення аплікатури, що позначена користуваем на інтерактивному грифі, на акорд. Аплікатура дублюється у одне з текстових полів для перевірки. У ще одне текстове поле виводиться назва акорду (рисунок 2.2). Якщо ж подібного акорду не існує, система сповістить про це користувача у цьому ж полі.

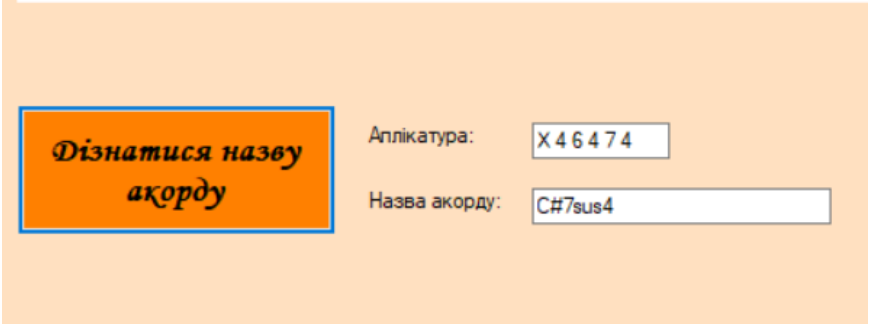

Рисунок 2.2 – Блок інтерфейсу кнопки конструктора акордів

Друга кнопка дозволяє працювати із зворотнім конструктором гітарних акордів. У одне текстове поле користувач вводить назву акорду. Після натискання на кнопку, додаток кодує назву акорду, та виводить його аплікатуру у загальноприйнятому вигляді у останнє текстове поле (рисунок 2.3). Числа у аплікатурі позначають струни та лади, на яких їх потрібно затиснути, щоб зіграти конкретний акорд.

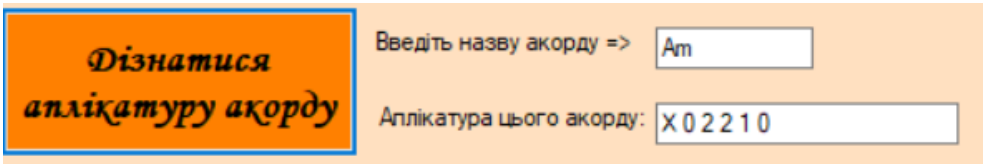

Рисунок 2.3 – Блок інтерфейсу кнопки зворотнього конструктора акордів

Третя кнопка відповідає за показ вікна, яке містить елемент «Кварто-квінтове коло», та знаходиться у правому верхньому куті головного вікна (рисунок 2.4).

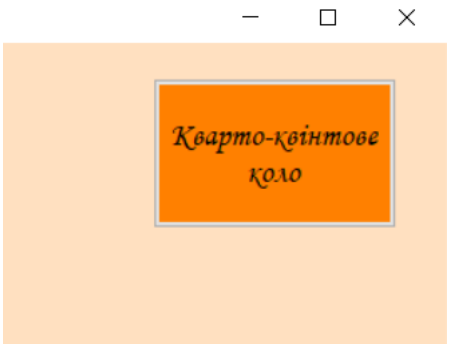

Рисунок 2.4 – Кнопка «Кварто-квінтове коло»

Кварто-квінтове коло – графічний елемент, що показує, які акорди можна поєднати, зігравши їх один за одним (рисунок 2.5). Один з необов`язкових, але дуже корисних інструментів для початківців у музиці.

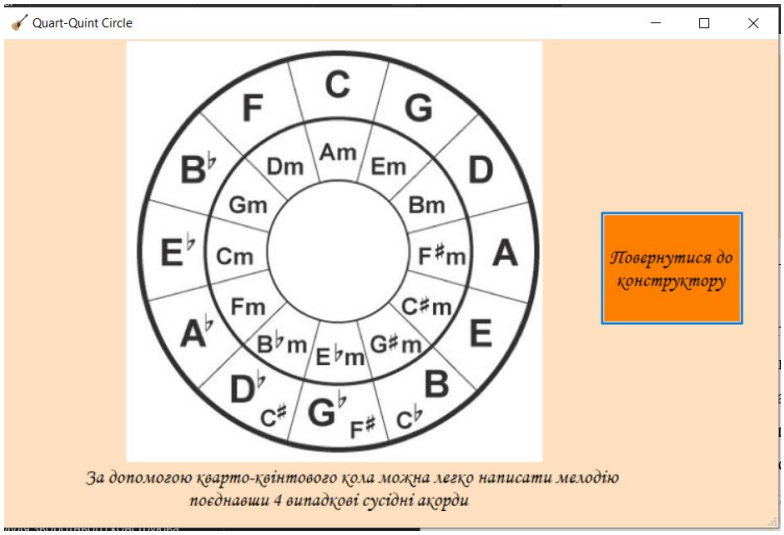

Рисунок 2.5 – Вікно «Кварто-квінтове коло»

Розробка графічного інтерфейсу додатку була проведена у середовищі розробки Microsoft Visual Studio 2019 з використанням технології Windows Forms.

Інтерфейс головного вікна додатку зображено на рисунку 2.6.

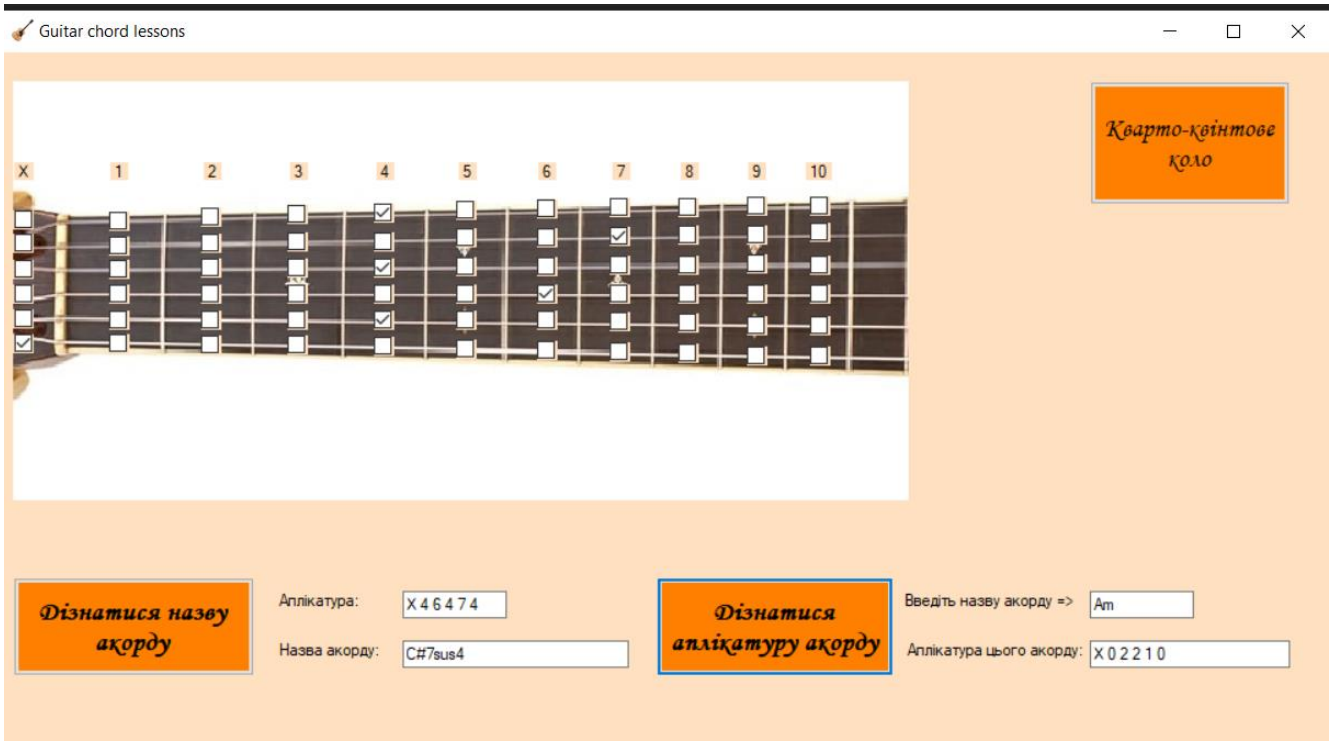

Рисунок 2.6 – Головне вікно програми

2.3 Розробка алгоритмів роботи додатку

Щоб розробити додаток для навчальної системи гри на гітарі потрібно спочатку розробити загальний алгоритм роботи додатку, а також алгоритми обробки вхідних та вихідних даних.

Загальний алгоритм роботи додатку має такі кроки: отримання команд або вхідних даних, обробка команд та даних, виконання поставленого завдання, обробка вихідних даних, вивід результату. Блок-схема алгоритму зображена на рисунку 2.7.

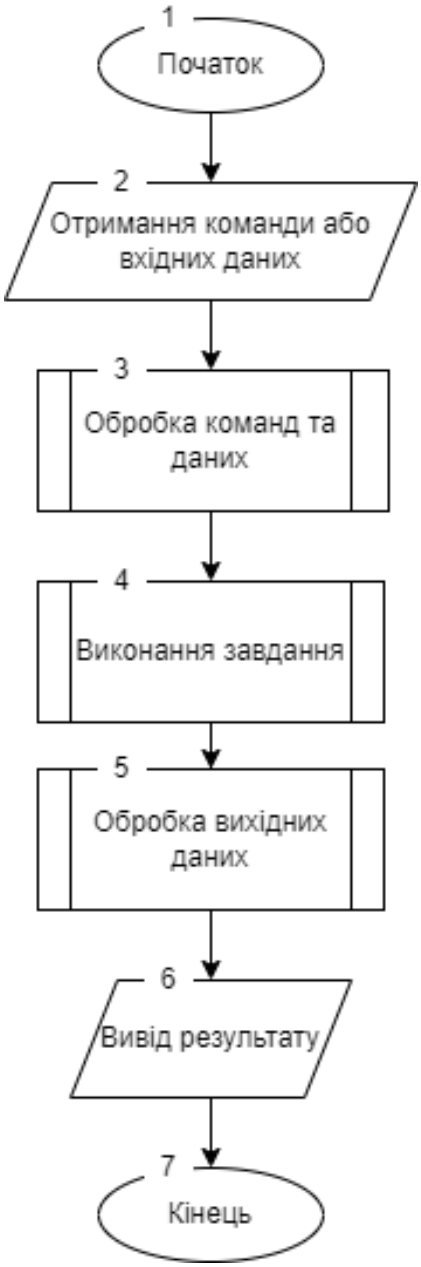

Рисунок 2.7 – Блок-схема загального алгоритму роботи

Наступний етап – розробка алгоритмів перетворення вхідних та вихідних даних. Для перетворення вхідних даних необхідно розбити текст на окремі слова. Далі для кожного слова перевіряємо, чи існує вектор для нього. Якщо існує, то додаємо у масив індекс вектору, якщо ні, додаємо до масиву індекс вектору, який відповідає за невідоме слово. На виході отримаємо масив індексів векторів, який подається на вхід до нейронної мережі. Блок-схема алгоритму зображена на рисунку 2.8.

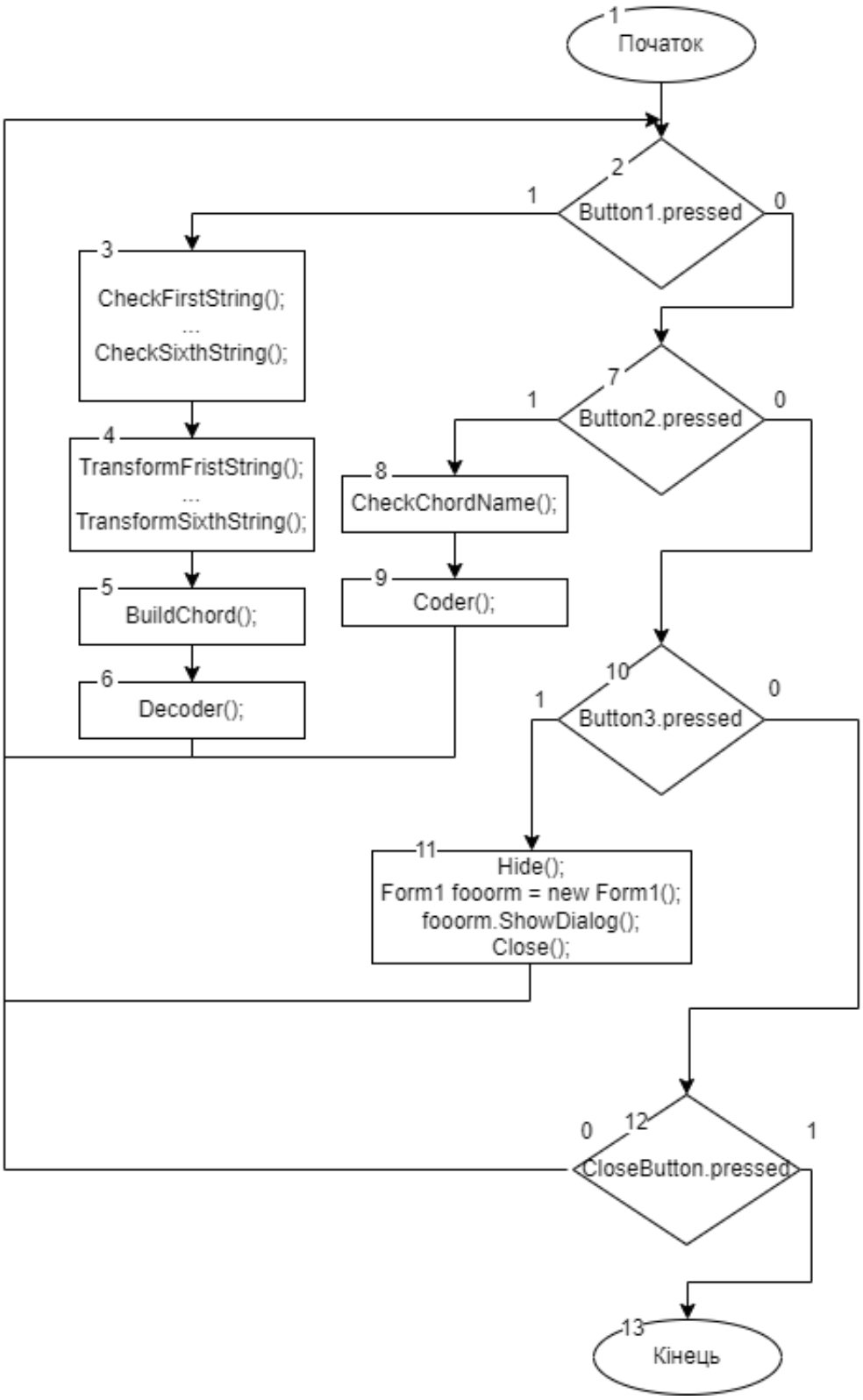

Рисунок 2.8 – Функціональна діаграма головного вікна додатку

При натискані третьої кнопки відкривається окреме вікно «кварто-квінтового кола». Алгоритм вікна продемонстровано на рисунку 2.9.

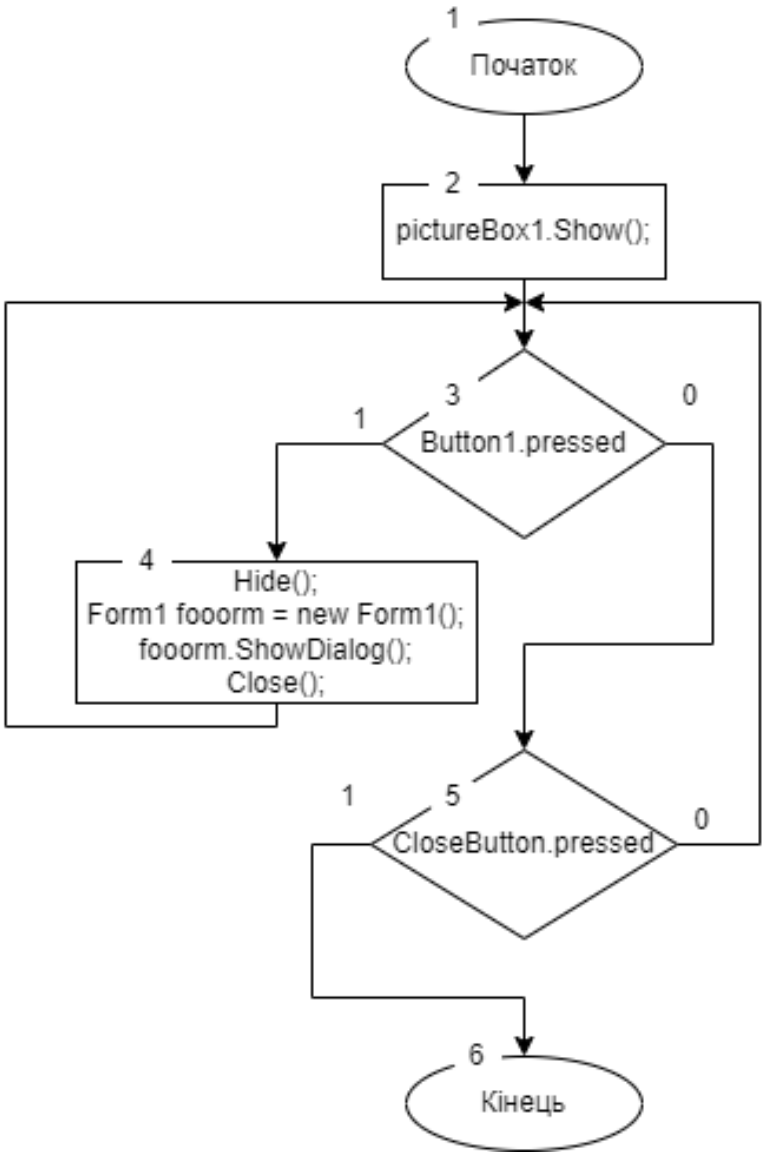

Рисунок 2.9 – Блок-схема алгоритму вінка «кварто-квінтове коло»

2.4 Висновки

В цьому розділі розділі було проаналізовано методи кодування даних, розроблено графічний інтерфейс додатку, розроблено загальний та функціональний алгоритм роботи головного вікна, а також алгоритм роботи додаткового вікна.

3 РОЗРОБКА МОДУЛІВ ЗВИЧАЙНОГО ТА ЗВОРОТНОГО КОНСТРУКТОРУ АКОРДІВ

3.1 Варіантний аналіз і обґрунтування вибору засобів для реалізації програмного засобу.

Вибір правильних технології для реалізації будь-якого програмного продукту є важливим етапом процесу розробки, адже правильна технологія дозволить значно полегшити процес реалізації. Для розробки програмного модуля було прийнято рішення використовувати дві мови програмуванн [15].

Мова програмування C#, у якій поєднуються потужність і гнучкість універсальних мов програмування з високою ефективністю виконавчого коду й можливістю безпосереднього доступу до апаратних ресурсів комп’ютера – одна з найкращих мов програмування [16].

C# створювалася як мова компонентного програмування, і в цьому одне з головних переваг мови, спрямоване на можливість повторного використання створених компонентів. З інших об'єктивних факторів [17]:

- C# створювався паралельно з каркасом Framework Net і повною мірою враховує всі його можливості - як FCL, так й CLR; C# є повністю об'єктно-орієнтованою мовою, де навіть типи, вбудовані в мову, представлені класами;
- C# є потужною об'єктною мовою з можливостями спадкування й універсалізації;
- C# є спадкоємцем мов C/C++, зберігаючи кращі риси цих популярних мов програмування;
- завдяки каркасу Framework .Net, що стали надбудовою над операційною системою, програмісти C# одержують ті ж переваги роботи з віртуальною машиною, що й програмісти Java. Ефективність коду навіть підвищується, оскільки виконавче середовище CLR являє собою

компілятор проміжної мови, у той час як віртуальна Java-машина є інтерпретатором байта-коду;

– потужна бібліотека каркасів підтримує зручність побудови різних типів додатків на C#, дозволяючи легко будувати Web-служби, інші види компонентів, досить просто зберігати й одержувати інформацію з бази даних й інших сховищ даних; реалізація, що сполучає побудову надійного й ефективного коду, є немаловажним чинником, що сприяє успіху C# [18].

Достойним конкурентом для C# є мова програмування Python. Python – високорівнева мова програмування загального призначення, яка орієнтована на підвищення продуктивності розробника та простоти коду за рахунок простого синтаксису.

Python в основному використовується в невеликих і великих онлайн-або офлайн-проектах, і він використовується для побудови графічного інтерфейсу, який розшифровується як графічний інтерфейс користувача, він також використовується для настільних додатків, тоді у нас є розробка ігор, тому Tkinter є стандартною бібліотекою графічного інтерфейсу для python, тому python у поєднанні Завдяки Tkinter це швидкий та простий спосіб створити графічний додаток та програми. Python також пропонує спеціальну основу для розробки ігор, яка є PYGAME.

Python підтримує орієнтоване програмування, воно дозволяє поліморфізм та успадкування. Користувачі Python отримують можливість користуватися спільними категоріями, таким чином код може бути багаторазовим і додатково забезпечувати механізм захисту шляхом абстрагування знань. Крім того, він звик розробляти прототипи, які модифікують користувача комп'ютера для прямого сканування та запису [19].

Python може обробляти велику кількість даних. Він підтримує паралельні обчислення або метод квадратної міри, готовий використовувати Python для цього, тому в python у нас в бібліотеці говориться як PYDOOP, щоб написати

програму MapReduce в python та інформацію про техніку, яка подарується з інтервалом в кластері HDFS. багато бібліотек люблять час доби та PySpark для гігантських процесів [20].

Зважаючи на всі переваги, було вирішено використати мову програмування C# для створення програмного додатку.

3.2 Розробка конструктору акордів

Інтерактивний гриф – основа для роботи конструктору акордів. Але потрібно вирішити ряд питань на рахунок його роботи.

Для створення інтерактивного грифу було використано середовище розробки Visual Studio 2019. Основні його елементи: Form, PictureBox, CheckBox, Button, Label та TextBox.

Алгоритм роботи наступний:

Спочатку користувач обирає потрібні йому елементи класу CheckBox, та відмічає їх на інтерактивному грифі. Потфм він натискає на кнопку «Дізнатися назву акорду», яка у свою чергу викликає функції (рисунок 3.1).

```
private void button1_Click(object sender, EventArgs e)
{
    CheckFirstString();
    CheckSecondString();
    CheckThirdString();
    CheckFourthString();
    CheckFifthString();
    CheckSixthString();
    TransformFirstString();
    TransformSecondString();
    TransformThirdString();
    TransformFourthString();
    TransformFifthString();
    TransformSixthString();
    BuildChord();
    Decoder();
}
```

Рисунок 3.1 – Лістинг коду кнопки-активатора конструктору

Після натискання кнопки «Дізнатися назву акорду» система перевіряє усі елементи CheckBox, та присвоює змінним, що відповідають за значення струн, ціле значення, що відповідає номеру останнього ладу, на якому затиснута струна.

Призначення відбувається у порядкуз першої по шосту струну (рисунки 3.2, 3.3, 3.4, 3.5, 3.6, 3.7). Якщо ж струна не зажата, або приглушена, змінна приймає значення «0» або «-1» відповідно.

```
public void CheckFirstString()
{
    firstString = 0;
    if (checkBox1.Checked)
    {
        firstString = 1;
    }
    if (checkBox2.Checked)
    {
        firstString = 2;
    }
    if (checkBox3.Checked)
    {
        firstString = 3;
    }
    if (checkBox4.Checked)
    {
        firstString = 4;
    }
    if (checkBox5.Checked)
    {
        firstString = 5;
    }
    if (checkBox6.Checked)
    {
        firstString = 6;
    }
```

Рисунок 3.2 – Фрагмент коду перевірки першої струни

```
public void CheckSecondString()
{
    secondString = 0;
    if (checkBox62.Checked)
    {
        secondString = -1;
    }
    if (checkBox11.Checked)
    {
        secondString = 1;
    }
    if (checkBox12.Checked)
    {
        secondString = 2;
    }
    if (checkBox13.Checked)
    {
        secondString = 3;
    }
    if (checkBox14.Checked)
    {
        secondString = 4;
    }
    if (checkBox15.Checked)
    {
        secondString = 5;
    }
    if (checkBox16.Checked)
    {
        secondString = 6;
```

Рисунок 3.3 – Фрагмент коду перевірки другої струни

```
public void CheckThirdString()
{
    thirdString = 0;
    if (checkBox63.Checked)
    {
        thirdString = -1;
    }
    if (checkBox21.Checked)
    {
        thirdString = 1;
    }
    if (checkBox22.Checked)
    {
        thirdString = 2;
    }
    if (checkBox23.Checked)
    {
        thirdString = 3;
    }
    if (checkBox24.Checked)
    {
        thirdString = 4;
    }
    if (checkBox25.Checked)
    {
        thirdString = 5;
    }
```

Рисунок 3.4 – Фрагмент коду перевірки третьої струни

```
public void CheckFourthString()
{
    fourthString = 0;
    if (checkBox64.Checked)
    {
        fourthString = -1;
    }
    if (checkBox31.Checked)
    {
        fourthString = 1;
    }
    if (checkBox32.Checked)
    {
        fourthString = 2;
    }
    if (checkBox33.Checked)
    {
        fourthString = 3;
    }
    if (checkBox34.Checked)
    {
        fourthString = 4;
    }
```

Рисунок 3.5 – Фрагмент коду перевірки четвертої струни

```
public void CheckFifthString()
{
    fifthString = 0;
    if (checkBox65.Checked)
    {
        fifthString = -1;
    }
    if (checkBox41.Checked)
    {
        fifthString = 1;
    }
    if (checkBox42.Checked)
    {
        fifthString = 2;
    }
    if (checkBox43.Checked)
    {
        fifthString = 3;
    }
    if (checkBox44.Checked)
    {
        fifthString = 4;
    }
    if (checkBox45.Checked)
    {
        fifthString = 5;
    }
```

Рисунок 3.6 – Фрагмент коду перевірки п`ятої струни

```
public void CheckSixthString()
{
    sixthString = 0;
    if (checkBox66.Checked)
    {
        sixthString = -1;
    }
    if (checkBox51.Checked)
    {
        sixthString = 1;
    }
    if (checkBox52.Checked)
    {
        sixthString = 2;
    }
    if (checkBox53.Checked)
    {
        sixthString = 3;
    }
    if (checkBox54.Checked)
    {
        sixthString = 4;
    }
    if (checkBox55.Checked)
    {
        sixthString = 5;
    }
    if (checkBox56.Checked)
    {
        sixthString = 6;
```

Рисунок 3.7 – Фрагмент функції перевірки шостої струни

Після присвоєння значень усім змінним, що відповідають за струни, дані конвертуються у формат string (рисунки 3.8, 3.9, 3.10), після чого поєднуються у змінній, що відповідає за аплікатуру акорду (рисунок 3.11).

```
public void TransformFirstString()
{

    if (firstString != -1)
    {
        firstBetween = firstString.ToString();
    }
    if (firstString == -1)
    {
        firstBetween = "X";
    }
}

public void TransformSecondString()
{
    if (secondString != -1)
    {
        secondBetween = secondString.ToString();
    }
    if (secondString == -1)
    {
        secondBetween = "X";
    }
}
```

Рисунок 3.8 – Функція перекодування інформації для першої та другої струни

```
public void TransformThirdString()
{
    if (thirdString != -1)
    {
        thirdBetween = thirdString.ToString();
    }
    if (thirdString == -1)
    {
        thirdBetween = "X";
    }
}

ссылка: 1
public void TransformFourthString()
{
    if (fourthString != -1)
    {
        fourthBetween = fourthString.ToString();
    }
    if (fourthString == -1)
    {
        fourthBetween = "X";
    }
}
```

Рисунок 3.9 - Функція перекодування інформації для третьої та четвертої струни

```
public void TransformFifthString()
{
    if (fifthString != -1)
    {
        fifthBetween = fifthString.ToString();
    }
    if (fifthString == -1)
    {
        fifthBetween = "X";
    }

}

ссылка: 1
public void TransformSixthString()
{
    if (sixthString != -1)
    {
        sixthBetween = sixthString.ToString();
    }
    if (sixthString == -1)
    {
        sixthBetween = "X";
    }
}
```

Рисунок 3.10 - Функція перекодування інформації для п`ятої та шостої струни

```
public void BuildChord()
{
    chordApplicature = sixthBetween;
    chordApplicature = chordApplicature + " " + fifthBetween;
    chordApplicature = chordApplicature + " " + fourthBetween;
    chordApplicature = chordApplicature + " " + thirdBetween;
    chordApplicature = chordApplicature + " " + secondBetween;
    chordApplicature = chordApplicature + " " + firstBetween;
    textBox1.Text = chordApplicature;
}
```

Рисунок 3.11 – Функція побудови акорду

Після перетворення даних та створення текстової аплікатури акорду, вона передається функції, що перевіряє її, та присвоює назву (рисунок 3.12).

```
public void Decoder()
{
    chordName = "No chord like this in database";
    if (chordApplicature.Equals("X 0 2 2 1 0"))
    {
        chordName = "Am";
    }
    if (chordApplicature.Equals("X 0 2 2 2 0"))
    {
        chordName = "A";
    }
    if (chordApplicature.Equals("X 0 2 2 0 0"))
    {
        chordName = "Asus2";
    }
    if (chordApplicature.Equals("X 0 2 2 3 0"))
    {
        chordName = "Asus4";
    }
    if (chordApplicature.Equals("X 0 2 0 1 0"))
    {
        chordName = "Am7";
    }
    if (chordApplicature.Equals("X 0 2 0 1 3"))
    {
        chordName = "Am7";
    }
    if (chordApplicature.Equals("X 0 2 0 2 0"))
```

Рисунок 3.12 – Фрагмент коду функції-декодеру конструктора гітарних акордів

Пройшовши усі стадії конструювання акорду, його аплікатура дублюється у один із елементів типу TextBox, а у інший записується назва акорду (рисунок 3.13).

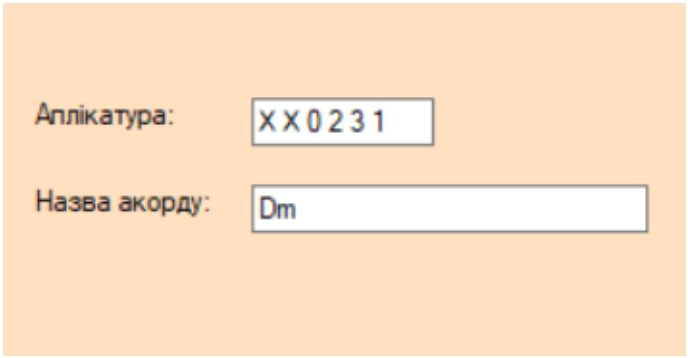

Рисунок 3.13 – Текстові поля, з інформацією про акорд

3.3 Розробка модуля зворотнього конструктору акордів

Наступним етапом розробки додатку є розробка зворотнього конструктора акордів.

Після натискання кнопки активуються функції зворотного конструктора (рисунок 3.14).

```
ссылка: 1
private void button2_Click(object sender, EventArgs e)
{
    CheckChordName();
    Coder();
}
```

Рисунок 3.14 – Лістинг коду кнопки зворотнього конструктора акордів

Спершу потрібно прийняти текстове значення із елементу типу TextBox, що було записане у вигляді «X X X X X X». Замість «X» повинні бути написані потрібні лади аплікатури, або залишити його, якщо струну потрібно заглушити при грі. Після цбого потрібно присвоїти це значення змінній (рисунок 3.15).

```
ссылка: 1
public void CheckChordName()
{
    chordApplicature = textBox3.Text;
}
ссылка: 1
```

Рисунок 3.15 – Лістинг коду функції прийому назви акорду

Змінна при пошуку буде порівнюватися з іншими записаними назвами акордів із різними базами, що відповідають за певну тональність, та можуть бути згруповані: A (рисунок 3.16), B (рисунок 3.17), C# (рисунок 3.18), D (рисунок 3.19), D# (рисунок 3.20), E (рисунок 3.21), F# (рисунок 3.22), G (рисунок 3.23), G# (рисунок 3.24).

```
}
ссылка: 1
public void Coder()
{
    chordName = "No chord like this in database";
    if (chordApplicature.Equals("Am"))
    {
        chordName = "X 0 2 2 1 0";
    }
    if (chordApplicature.Equals("A"))
    {
        chordName = "X 0 2 2 2 0";
    }
    if (chordApplicature.Equals("Asus2"))
    {
        chordName = "X 0 2 2 0 0";
    }
    if (chordApplicature.Equals("Asus4"))
    {
        chordName = "X 0 2 2 3 0";
    }
    if (chordApplicature.Equals("Am7"))
    {
        chordName = "X 0 2 0 1 0";
    }
    if (chordApplicature.Equals("Am7"))
    {
        chordName = "X 0 2 0 1 3";
    }
    if (chordApplicature.Equals("A7"))
    {
        chordName = "X 0 2 0 2 0";
```

Рисунок 3.16 – Фрагмент коду функції кодування акордів з базою «A» зворотнього конструктора акордів

```
if (chordApplicature.Equals("Bm"))
{
    chordName = "X 2 4 4 3 2";
}
if (chordApplicature.Equals("B"))
{
    chordName = "X 2 4 4 4 2";
}
if (chordApplicature.Equals("Bsus2"))
{
    chordName = "X 2 4 4 2 2";
}
if (chordApplicature.Equals("Bsus4"))
{
    chordName = "X 2 4 4 5 2";
}
if (chordApplicature.Equals("B7"))
{
    chordName = "X 2 4 2 4 2";
}
if (chordApplicature.Equals("B7"))
{
    chordName = "X 2 4 2 4 5";
}
if (chordApplicature.Equals("B7"))
{
    chordName = "X 2 1 2 0 2";
}
if (chordApplicature.Equals("B7sus4"))
{
```

Рисунок 3.17 – Фрагмент коду функції кодування акордів з базою «B» зворотнього конструктора акордів

```
if (chordApplicature.Equals("C#"))
{
    chordName = "X 4 6 6 6 4";
}
if (chordApplicature.Equals("C#m"))
{
    chordName = "X 4 6 6 5 4";
}
if (chordApplicature.Equals("C#sus2"))
{
    chordName = "X 4 6 6 4 4";
}
if (chordApplicature.Equals("C#sus4"))
{
    chordName = "X 4 6 6 7 4";
}
if (chordApplicature.Equals("C#7"))
{
    chordName = "X 4 6 4 6 4";
}
if (chordApplicature.Equals("C#7"))
{
    chordName = "X 4 6 4 6 7";
}
if (chordApplicature.Equals("C#m7"))
{
    chordName = "X 4 6 4 5 4";
}
if (chordApplicature.Equals("C#m7"))
{
    chordName = "X 4 6 4 5 7";
```

Рисунок 3.18 – Фрагмент коду функції кодування акордів з базою «C#» зворотнього конструктора акордів

```
}
if (chordApplicature.Equals("Dm"))
{
    chordName = "X 5 7 7 6 5";
}
if (chordApplicature.Equals("Dmaj7b5"))
{
    chordName = "X X 0 2 3 0";
}
if (chordApplicature.Equals("Dmaj7b5"))
{
    chordName = "X 0 0 2 3 0";
}
if (chordApplicature.Equals("Dsus2"))
{
    chordName = "X 5 7 7 5 5";
}
if (chordApplicature.Equals("Dsus4"))
{
    chordName = "X 5 7 7 8 5";
}
if (chordApplicature.Equals("D7"))
{
    chordName = "X 5 7 5 7 5";
```

Рисунок 3.19 – Фрагмент коду функції кодування акордів з базою «D» зворотнього конструктора акордів

```
if (chordApplicature.Equals("D#"))
{
    chordName = "X 6 8 8 8 6";
}
if (chordApplicature.Equals("D#m"))
{
    chordName = "X X 1 3 4 2";
}
if (chordApplicature.Equals("D#m"))
{
    chordName = "X 6 8 8 7 6";
}
if (chordApplicature.Equals("D#sus2"))
{
    chordName = "X 6 8 8 6 6";
}
if (chordApplicature.Equals("D#sus4"))
{
    chordName = "X 6 8 8 9 6";
}
if (chordApplicature.Equals("D#7"))
{
    chordName = "X 6 8 6 8 6";
}
if (chordApplicature.Equals("D#7"))
{
    chordName = "X 6 8 6 7 9";
}
```

Рисунок 3.20 – Фрагмент коду функції кодування акордів з базою «D» зворотнього конструктора акордів

```
if (chordApplicature.Equals("E"))
{
    chordName = "0 2 2 1 0 0";
}
if (chordApplicature.Equals("Em"))
{
    chordName = "0 2 2 0 0 0";
}
if (chordApplicature.Equals("Esus4"))
{
    chordName = "0 2 2 2 0 0";
}
if (chordApplicature.Equals("E7"))
{
    chordName = "0 2 0 1 3 0";
}
if (chordApplicature.Equals("Em7"))
{
    chordName = "0 2 0 0 3 0";
}
if (chordApplicature.Equals("E6"))
{
    chordName = "0 2 2 1 1 0";
}
if (chordApplicature.Equals("Em6"))
{
    chordName = "0 2 2 0 1 0";
}
```

Рисунок 3.21 – Фрагмент коду функції кодування акордів з базою «E» зворотнього конструктора акордів

```
if (chordApplicature.Equals("F#"))
{
    chordName = "2 4 4 3 2 2";
}
if (chordApplicature.Equals("F#m"))
{
    chordName = "2 4 4 2 2 2";
}
if (chordApplicature.Equals("F#sus4"))
{
    chordName = "2 4 4 4 2 2";
}
if (chordApplicature.Equals("F#7"))
{
    chordName = "2 4 2 3 2 2";
}
if (chordApplicature.Equals("F#7"))
{
    chordName = "2 4 2 3 5 2";
}
if (chordApplicature.Equals("F#m7"))
{
    chordName = "2 4 2 2 2 2";
}
if (chordApplicature.Equals("F#m7"))
{
    chordName = "2 4 2 2 5 2";
}
if (chordApplicature.Equals("F#6"))
{
```

Рисунок 3.22 – Фрагмент коду функції кодування акордів з базою «F#» зворотнього конструктора акордів

```
if (chordApplicature.Equals("G"))
{
    chordName = "3 2 0 0 3 3";
}
if (chordApplicature.Equals("G(III)"))
{
    chordName = "3 5 5 4 3 3";
}
if (chordApplicature.Equals("Gm"))
{
    chordName = "3 5 5 3 3 3";
}
if (chordApplicature.Equals("Gsus4"))
{
    chordName = "3 5 5 5 3 3";
}
if (chordApplicature.Equals("G7"))
{
    chordName = "3 5 3 4 3 3";
}
if (chordApplicature.Equals("G7"))
{
    chordName = "3 5 3 4 6 3";
}
if (chordApplicature.Equals("Gm7"))
{
    chordName = "3 5 3 3 3 3";
}
if (chordApplicature.Equals("Gm7"))
{
```

Рисунок 3.23 – Фрагмент коду функції кодування акордів з базою «G» зворотнього конструктора акордів

```
if (chordApplicature.Equals("G#"))
{
    chordName = "4 6 6 5 4 4";
}
if (chordApplicature.Equals("G#m"))
{
    chordName = "4 6 6 4 4 4";
}
if (chordApplicature.Equals("G#sus4"))
{
    chordName = "4 6 6 6 4 4";
}
if (chordApplicature.Equals("G#7"))
{
    chordName = "4 6 4 5 4 4";
}
if (chordApplicature.Equals("G#7"))
{
    chordName = "4 6 4 5 7 4";
}
if (chordApplicature.Equals("G#m7"))
{
    chordName = "4 6 4 4 4 4";
}
if (chordApplicature.Equals("G#m7"))
{
    chordName = "4 6 4 4 7 4";
}
```

Рисунок 3.24 – Фрагмент коду функції кодування акордів з базою «G» зворотнього конструктора акордів

У разі знаходження потрібної назви, у сусіднє поле типу TextBox виводиться аплікатура акорду (рисунок 3.25).

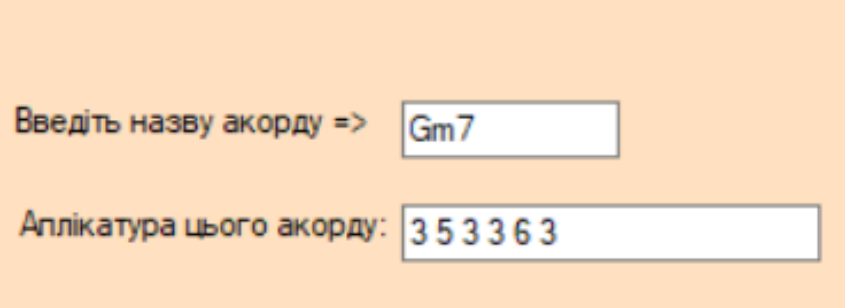

Рисунок 3.25 – Текстові поля з інформацією про акорд

3.4 Висновки

У третьому розділі було обґрунтовано вибір мови програмування та технологій, які будуть використовуватися при розробці програмного продукту та наведено основні їх переваги. У результаті аналізу було обрано мову програмування C# для написання коду модулів програмного додатку та середовище розробки Microsoft Visual Studio 2019 для розробки графічного інтерфейсу користувача.

Було проведено розробку модулів конструктору гітарних акордів та зворотного конструктору гітарних акордів та пов`язано їх із графічним інтерфейсом користувача.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Тестування додатку

Тестування додатку – це процес, під час якого перевіряється його здібність до стабільної роботи, зручність експлуатації його користувачами, а також проводиться перевірка на наявність різних багів та помилок.

Під час запуску програмного додатку користувач потрапляє до головного вікна (рисунок 4.1).

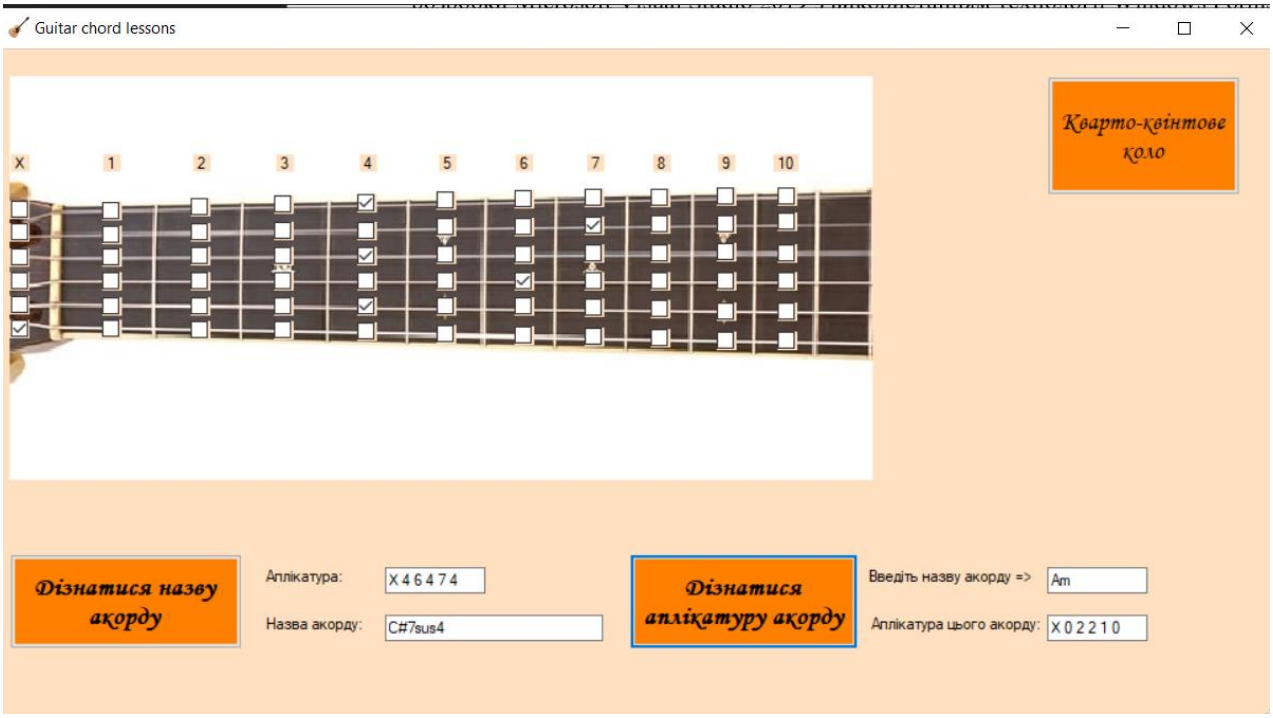

Рисунок 4.1 – Головне вікно програми

Для початку роботи із конструктором гітарних акордів користувачу потрібно позначити бажану аплікатуру на інтерактивному грифі, обравши потрібні елементи CheckBox (рисунок 4.2). Якщо користувач обере декілька елементів на одній струні, то по привилах гри на гітарі, затиснутим буде вважатися саме останній затиснутий лад (рисунок 4.3).

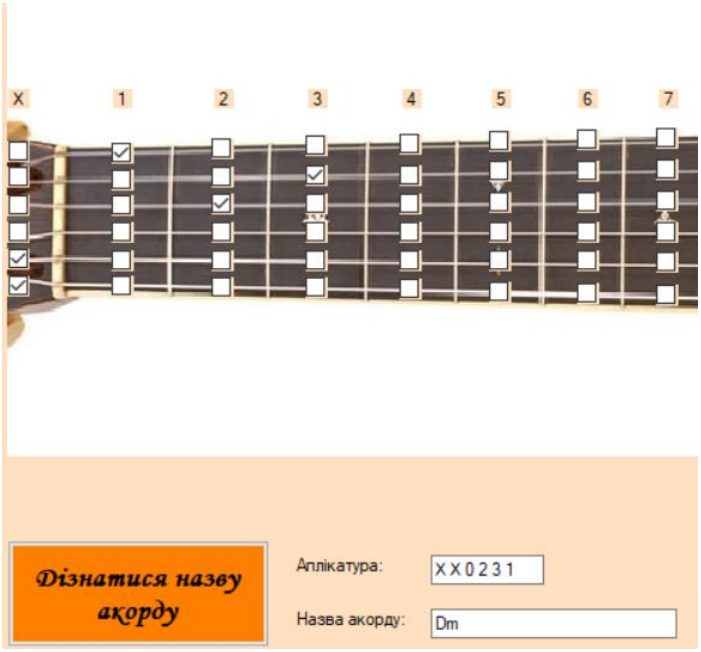

Рисунок 4.2 – Приклад затиснутого акорду на інтерактивному грифі

Рисунок 4.3 – Приклад затиснутого акорду із кількома обраними об`єктами на одній струні

У випадку, якщо затиснутого акорду не існує, система виведе повідомлення у текстове поле із назвою акорду про відсутність його у її базі даних сповістивши про це користувача (рисунок 4.4).

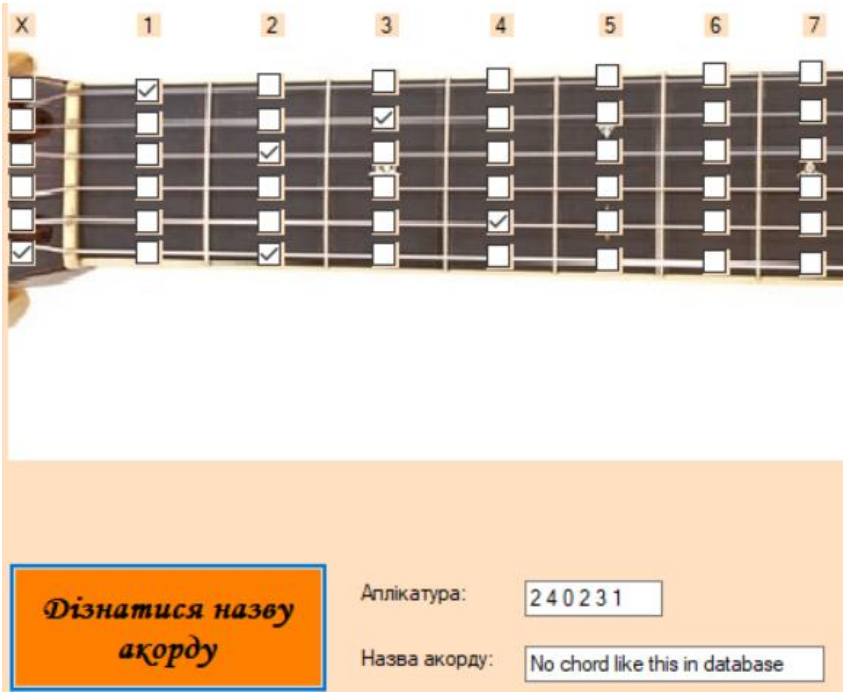

Рисунок 4.4 – Приклад сповіщення про відсутність акорду у базі

Для роботи із зворотнім конструктором акордів користувачу потрібно записати назву акорду у текстове поле, що призначене для цього. В залежності від наявності такого акорду у базі даних програми, вона виведе його аплікатуру (рисунок 4.5) або сповіщення про його відсутність (рисунок 4.6).

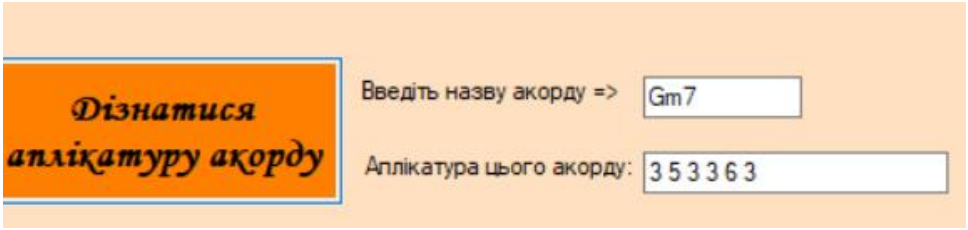

Рисунок 4.5 – Виведення аплікатури успішно знайденого акорду

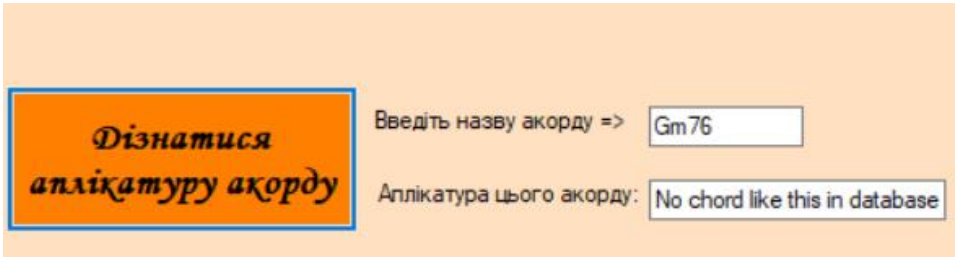

Рисунок 4.6 – Сповіщення про відсутність акорду у базі даних програми

При натисканні на кнопку «Каврто-квінтове коло» додаток відкриває нове вікно з одноіменним інструментом (рисунок 4.7).

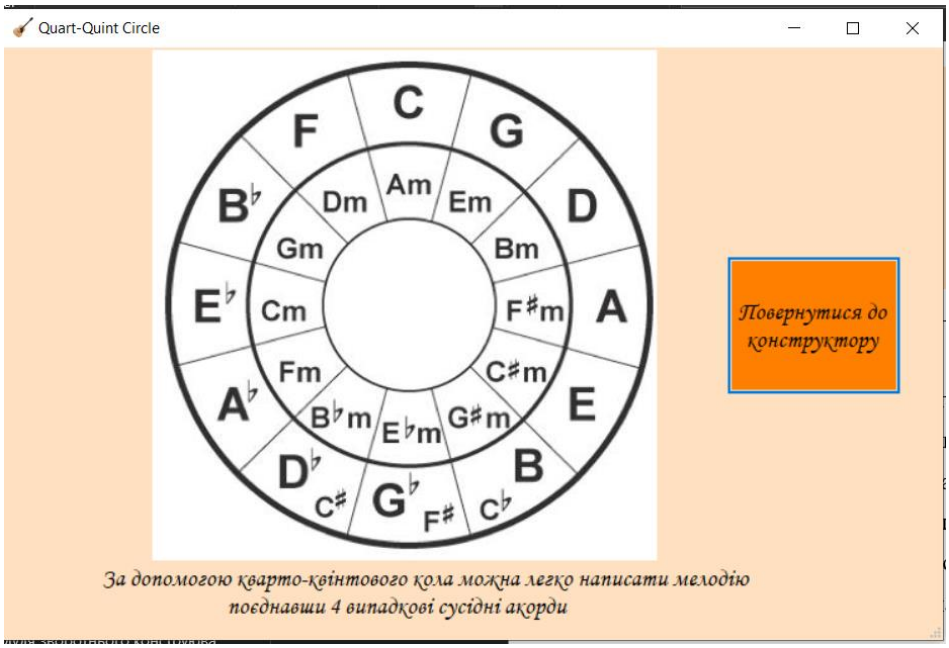

Рисунок 4.7 – Вікно «Кварто-квінтове коло»

4.2 Розробка інструкції користувача

Інструкція користувача передбачає визначення технічних вимог для запуску програмного продукту. Деталі щодо мінімальної та рекомендованої конфігурації серверного комп’ютера можна знайти в таблицях 4.1 та 4.2.

Таблиця 4.1 – Мінімальна конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з кількістю фізичних ядер 2 та тактовою частотою 1.7 ГГц
Об’єм оперативної пам’яті	1 ГБ для 32-розрядної системи і 2 ГБ для 64-разрядної системи
Тип жорсткого диску	HDD
Операційна система	Windows 8.1

Таблиця 4.2 – Рекомендована конфігурація:

Тип процесора	32-розрядний (x86) або 64-розрядний (x64) процесор з кількістю фізичних ядер 2 та тактовою частотою 2.1 ГГц
Об’єм оперативної пам’яті	2 ГБ для 32-розрядної системи і 4 ГБ для 64-разрядної системи
Тип жорсткого диску	SSD
Операційна система	Windows 10

Для коректної роботи користувачу потрібно перевірити наявність на своєму комп`ютері таких елементів, як встановити Microsoft Visual C++ 2015-2022 Redistributable (x64) або Microsoft Visual C++ 2015-2022 Redistributable (x86) в залежності від типу розрядності операційної системи. Швидко і зручно зробити це можна за допомогою інструменту «програми і компоненти», що знаходиться у «панелі керування». Якщо немає таких встановлених програмних засобів, потрібно встановити їх за допомогою офіційного сервісу або веб-сайту компанії Microsoft.

4.3 Висновки

У четвертому розділі було протестовано програмний додаток для навчальної системи гри на гітарі. Було підтверджено працездатність усіх блоків програмного продукту, та перевірено їх на відсутність багів або помилок. Також було визначено мінімальну та рекомендовану конфігурацію для запуску додатку та роботи із ним. Написано інструкцію користувача по експлуатації додатку та інструкцію із встановленню сторонніх додаткових компонентів для його коректної роботи.

ВИСНОВКИ

У процесі проходження практики було розроблено програмний додаток для навчальної системи гри на гітарі. Для розробки було використано середовище програмування Visual Studio 2019.

Було проаналізовано актуальність теми. Розглянуто основні аналоги, визначено їх особливості та недоліки і розроблено порівняння з власним програмним продуктом.

У результаті аналізу обрано мови програмування C# для розробки усіх блоків програмного додатку.

У результаті проходження практики було зроблено наступне:

- розроблено метод та алгоритм роботи інтерактивного грифу гітари;
- розроблено алгоритм роботи конструктора акордів;
- розроблено алгоритм роботи зворотнього конструктора акордів;
- розроблено зручний та зрозумілий графічний інтерфейс;
- розроблено програмний додаток для навчальної системи гри на гітарі;

Перед створенням програмного продукту було розроблено схеми загального алгоритму роботи додатку, обробки вхідних та вихідних даних, розроблено його структуру.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Засоби для налаштування музичних інструментів : [Електронний ресурс] – режим доступу до ресурсу: https://yousician.com/guitartuna
2. Конструктор гітарних акордів : [Електронний ресурс] – режим доступу до ресурсу: https://chords.top/generator/
3. Інтернет бібліотека пісень: [Електронний ресурс] – режим доступу до ресурсу: https://mychords.net/uk/gens
4. Зворотній конструктор гітарних акордів : [Електронний ресурс] – режим доступу до ресурсу: http://u-rock.com.ua/chord-generator.html
5. C# Створення бібліотек : [Електронний ресурс] – режим доступу до ресурсу: https://metanit.com/sharp/tutorial/3.46.php
6. Іан Соммервіллем Інженерія програмного забезпечення = Software Engineering. — 6-е вид. — М.: «Вильямс», 2002. — С. 642.
7. Джек Грінфілд, Кіт Шорт, Стів Кук, Стюарт Кент, Джон Крупи Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools. — М.: «Діалектика», 2006. — С. 592.
8. Комп’ютерна техніка та інформаційні технології : навчальний посібник 2-е вид. / Борис Погріщук, Ю. Паночишин. – Київ : Знання – 463 ст.
9. Стандарти кодування тексту : [Електронний ресурс] – режим доступу до ресурсу: https://sites.google.com/site/navcalnijsajtdlastudentivvnz/teoreticni-osnovi-informatiki/lekciie-17-18/lekcia_3_toi
10. ASCII особливості використання : [Електронний ресурс] – режим доступу до ресурсу: https://www.wikiwand.com/uk/ASCII
11. Захарова В. І., Філіпова В. Я. Основи інформаційно-аналітичної діяльності — К. «Центр учбової літератури», 2013. — 336 с
12. Юнікод як стандарт кодування : [Електронний ресурс] – режим доступу до ресурсу: https://support.microsoft.com/uk-ua/office/вибір-кодування-тексту-під-час-відкривання-та-зберігання-файлів-60d59c21-88b5-4006-831c

13. Алгоритми. Побудова і аналіз : пер. с англ. / Т. Кормен [і ін.]. - 2-е вид. - М. ; СП. ; К. : Вільямс, 2007. - 1296 с.
14. The Humane Interface: New Directions for Designing Interactive Systems: навч. посіб. / Addison-Wesley Professional Publisher; 2000. – 233 ст.
15. Лабор В. В. C#: Створення додатків для Windows/В. В. Лабор.— Мн.: Харвест, 2003. - 384 с.
16. Костриба О.В. Основи програмування. Частина 1. Visual C# Express Edition / О.В. Костриба. – Білогір'я, 2008. – 74 с.
17. Коноваленко І.В. Програмування мовою C# 7.0 : навчальний посібник / Коноваленко І.В., Марущак П.О., Савків В.Б. – Тернопіль : ТНТУ імені Івана Пулюя 2017 – 300 с.
18. Переваги мови програмуваання C# : [Електронний ресурс] – режим доступу до ресурсу: https://dou.ua/lenta/articles/pros-and-cons-of-dotnet/?hl=
19. Факти мови програмування Python: [Електронний ресурс] – режим доступу до ресурсу: http://www.itschool.vn.ua/interesting-python/
20. Переваги та недоліки мови програмування Python : [Електронний ресурс] – режим доступу до ресурсу: https://uk.education-wiki.com/6821261-advantages-of-python

Додаток А – Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
25 березня 2022 р.

Технічне завдання
на бакалаврську дипломну роботу «Розробка програмного забезпечення для навчальної системи гри на гітарі»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник бакалаврської дипломної роботи:
_____ к.т.н., доц. Черноволик Г.О.
"31" березня 2022 р.
Виконав:
_____студент гр. 2ПІ-18б Бодовський В.Д.
"31" березня 2022 р.

Вінниця – 2022 року

1. Найменування та галузь застосування

Бакалаврська дипломна робота: «Розробка програмного забезпечення для навчальної системи гри на гітарі».

Галузь застосування – десктоп-технології.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення – протокол № 12 від « 7 » лютого 2022 р.

3. Мета та призначення розробки.

Метою роботи є удосконалення процесу звичайного та зворотного конструювання гітарних акордів, а також позбавлення цих модулів від залежності до підключенні до інтернет-мережі.

Призначення роботи – автоматизації процесу конструювання та зворотного конструювання гітарних акордів.

4. Вхідні дані для проведення НДР

1. Алгоритми. Побудова і аналіз : пер. с англ. / Т. Кормен [і ін.]. - 2-е вид. - М. ; СП. ; К. : Вільямс, 2007. - 1296 с.
2. Коноваленко І.В. Програмування мовою C# 7.0 : навчальний посібник / Коноваленко І.В., Марущак П.О., Савків В.Б. – Тернопіль : ТНТУ імені Івана Пулюя 2017 – 300 с.
3. Комп’ютерна техніка та інформаційні технології : навчальний посібник 2-е вид. / Борис Погріщук, Ю. Паночишин. – Київ : Знання – 463 ст.
4. The Humane Interface: New Directions for Designing Interactive Systems: навч. посіб. / Addison-Wesley Professional Publisher; 2000. – 233 ст.

5. Технічні вимоги

Вхідні дані – аплікатура акорду на інтерактивному грифі; вихідні дані – графічний інтерфейс, що надає інформацію про конкретний акорд.

6. Конструктивні вимоги.

Конструкція програмного додатку повинна відповідати ергономічним та естетичним вимогам, повинна бути зручною в керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до бакалаврської дипломної роботи;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

Додаток Б – Протокол перевірки

ПРОТОКОЛ
ПЕРЕВІРКИ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Розробка програмного забезпечення для навчальної системи гри на гітарі

Тип роботи: БДР

Підрозділ : кафедра програмного забезпечення, ФІТКІ

Науковий керівник: Черноволик Г.О.

Оригінальність	86
Схожість	14

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

 Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

 Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

5

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Бодовський В.Д.

Керівник роботи _____ Черноволик Г. О.

Додаток В – Лістинг програми

C#

Form1.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;


namespace DyplomProgram
{
    public partial class Form1 : Form
    {
        int firstString = 0;
        int secondString = 0;
        int thirdString = 0;
        int fourthString = 0;
        int fifthString = 0;
        int sixthString = 0;
        string firstBetween;
        string secondBetween;
        string thirdBetween;
        string fourthBetween;
        string fifthBetween;
        string sixthBetween;
        string chordName;
        string chordApplicature;
        public void CheckFirstString()
        {
            firstString = 0;
            if (checkBox1.Checked)
            {
                firstString = 1;
            }
            if (checkBox2.Checked)
```

```
    {
        firstString = 2;
    }
    if (checkBox3.Checked)
    {
        firstString = 3;
    }
    if (checkBox4.Checked)
    {
        firstString = 4;
    }
    if (checkBox5.Checked)
    {
        firstString = 5;
    }
    if (checkBox6.Checked)
    {
        firstString = 6;
    }
    if (checkBox7.Checked)
    {
        firstString = 7;
    }
    if (checkBox8.Checked)
    {
        firstString = 8;
    }
    if (checkBox9.Checked)
    {
        firstString = 9;
    }
    if (checkBox10.Checked)
    {
        firstString = 10;
    }
    if (checkBox61.Checked)
    {
        firstString = -1;
    }
}

public void CheckSecondString()
{
```

```
secondString = 0;
if (checkBox62.Checked)
{
    secondString = -1;
}
if (checkBox11.Checked)
{
    secondString = 1;
}
if (checkBox12.Checked)
{
    secondString = 2;
}
if (checkBox13.Checked)
{
    secondString = 3;
}
if (checkBox14.Checked)
{
    secondString = 4;
}
if (checkBox15.Checked)
{
    secondString = 5;
}
if (checkBox16.Checked)
{
    secondString = 6;
}
if (checkBox17.Checked)
{
    secondString = 7;
}
if (checkBox18.Checked)
{
    secondString = 8;
}
if (checkBox19.Checked)
{
    secondString = 9;
}
if (checkBox20.Checked)
{
```

```
            secondString = 10;
        }

    }

    public void CheckThirdString()
    {
        thirdString = 0;
        if (checkBox63.Checked)
        {
            thirdString = -1;
        }
        if (checkBox21.Checked)
        {
            thirdString = 1;
        }
        if (checkBox22.Checked)
        {
            thirdString = 2;
        }
        if (checkBox23.Checked)
        {
            thirdString = 3;
        }
        if (checkBox24.Checked)
        {
            thirdString = 4;
        }
        if (checkBox25.Checked)
        {
            thirdString = 5;
        }
        if (checkBox26.Checked)
        {
            thirdString = 6;
        }
        if (checkBox27.Checked)
        {
            thirdString = 7;
        }
        if (checkBox28.Checked)
        {
            thirdString = 8;
```

```
    }
    if (checkBox29.Checked)
    {
        thirdString = 9;
    }
    if (checkBox30.Checked)
    {
        thirdString = 10;
    }

}

public void CheckFourthString()
{
    fourthString = 0;
    if (checkBox64.Checked)
    {
        fourthString = -1;
    }
    if (checkBox31.Checked)
    {
        fourthString = 1;
    }
    if (checkBox32.Checked)
    {
        fourthString = 2;
    }
    if (checkBox33.Checked)
    {
        fourthString = 3;
    }
    if (checkBox34.Checked)
    {
        fourthString = 4;
    }
    if (checkBox35.Checked)
    {
        fourthString = 5;
    }
    if (checkBox36.Checked)
    {
        fourthString = 6;
    }
```

```
    if (checkBox37.Checked)
    {
        fourthString = 7;
    }
    if (checkBox38.Checked)
    {
        fourthString = 8;
    }
    if (checkBox39.Checked)
    {
        fourthString = 9;
    }
    if (checkBox40.Checked)
    {
        fourthString = 10;
    }

}

public void CheckFifthString()
{
    fifthString = 0;
    if (checkBox65.Checked)
    {
        fifthString = -1;
    }
    if (checkBox41.Checked)
    {
        fifthString = 1;
    }
    if (checkBox42.Checked)
    {
        fifthString = 2;
    }
    if (checkBox43.Checked)
    {
        fifthString = 3;
    }
    if (checkBox44.Checked)
    {
        fifthString = 4;
    }
    if (checkBox45.Checked)
```

```
    {
        fifthString = 5;
    }
    if (checkBox46.Checked)
    {
        fifthString = 6;
    }
    if (checkBox47.Checked)
    {
        fifthString = 7;
    }
    if (checkBox48.Checked)
    {
        fifthString = 8;
    }
    if (checkBox49.Checked)
    {
        fifthString = 9;
    }
    if (checkBox50.Checked)
    {
        fifthString = 10;
    }

}

public void CheckSixthString()
{
    sixthString = 0;
    if (checkBox66.Checked)
    {
        sixthString = -1;
    }
    if (checkBox51.Checked)
    {
        sixthString = 1;
    }
    if (checkBox52.Checked)
    {
        sixthString = 2;
    }
    if (checkBox53.Checked)
    {
```

```
            sixthString = 3;
        }
        if (checkBox54.Checked)
        {
            sixthString = 4;
        }
        if (checkBox55.Checked)
        {
            sixthString = 5;
        }
        if (checkBox56.Checked)
        {
            sixthString = 6;
        }
        if (checkBox57.Checked)
        {
            sixthString = 7;
        }
        if (checkBox58.Checked)
        {
            sixthString = 8;
        }
        if (checkBox59.Checked)
        {
            sixthString = 9;
        }
        if (checkBox60.Checked)
        {
            sixthString = 10;
        }

    }

    public void TransformFirstString()
    {

        if (firstString != -1)
        {
            firstBetween = firstString.ToString();
        }
        if (firstString == -1)
        {
            firstBetween = "X";
```

```
    }
}

public void TransformSecondString()
{
    if (secondString != -1)
    {
        secondBetween = secondString.ToString();
    }
    if (secondString == -1)
    {
        secondBetween = "X";
    }
}

public void TransformThirdString()
{
    if (thirdString != -1)
    {
        thirdBetween = thirdString.ToString();
    }
    if (thirdString == -1)
    {
        thirdBetween = "X";
    }
}

public void TransformFourthString()
{
    if (fourthString != -1)
    {
        fourthBetween = fourthString.ToString();
    }
    if (fourthString == -1)
    {
        fourthBetween = "X";
    }
}

public void TransformFifthString()
{
    if (fifthString != -1)
    {
```

```
            fifthBetween = fifthString.ToString();
        }
        if (fifthString == -1)
        {
            fifthBetween = "X";
        }

    }

    public void TransformSixthString()
    {
        if (sixthString != -1)
        {
            sixthBetween = sixthString.ToString();
        }
        if (sixthString == -1)
        {
            sixthBetween = "X";
        }
    }

    public void BuildChord()
    {
        chordApplicature = sixthBetween;
        chordApplicature = chordApplicature + " " + fifthBetween;
        chordApplicature = chordApplicature + " " + fourthBetween;
        chordApplicature = chordApplicature + " " + thirdBetween;
        chordApplicature = chordApplicature + " " + secondBetween;
        chordApplicature = chordApplicature + " " + firstBetween;
        textBox1.Text = chordApplicature;
    }

    public void CheckChordName()
    {
        chordApplicature = textBox3.Text;
    }
    public void Coder()
    {
        chordName = "No chord like this in database";
        if (chordApplicature.Equals("Am"))
        {
            chordName = "X 0 2 2 1 0";
        }
```

```
if (chordApplicature.Equals("A"))
{
    chordName = "X 0 2 2 2 0";
}
if (chordApplicature.Equals("Asus2"))
{
    chordName = "X 0 2 2 0 0";
}
if (chordApplicature.Equals("Asus4"))
{
    chordName = "X 0 2 2 3 0";
}
if (chordApplicature.Equals("Am7"))
{
    chordName = "X 0 2 0 1 0";
}
if (chordApplicature.Equals("Am7"))
{
    chordName = "X 0 2 0 1 3";
}
if (chordApplicature.Equals("A7"))
{
    chordName = "X 0 2 0 2 0";
}
if (chordApplicature.Equals("A7"))
{
    chordName = "X 0 2 0 2 3";
}
if (chordApplicature.Equals("A6"))
{
    chordName = "X 0 2 2 2 2";
}
if (chordApplicature.Equals("Am/C"))
{
    chordName = "X 3 2 2 1 0";
}
if (chordApplicature.Equals("Am(V)"))
{
    chordName = "5 7 7 5 5 5";
}
if (chordApplicature.Equals("A(V)"))
{
    chordName = "5 7 7 6 5 5";
```

```
}
if (chordApplicature.Equals("Am7(V)"))
{
    chordName = "5 7 5 5 8 5";
}
if (chordApplicature.Equals("Asus4(V)"))
{
    chordName = "5 7 7 7 5 5";
}
if (chordApplicature.Equals("A7(V)"))
{
    chordName = "5 7 5 6 8 5";
}
if (chordApplicature.Equals("Am6(V)"))
{
    chordName = "5 7 7 5 7 5";
}
if (chordApplicature.Equals("A6(II)"))
{
    chordName = "2 4 2 2 2 2";
}
if (chordApplicature.Equals("A#m"))
{
    chordName = "X 1 3 3 2 1";
}
if (chordApplicature.Equals("A#"))
{
    chordName = "X 1 3 3 3 1";
}
if (chordApplicature.Equals("A#sus2"))
{
    chordName = "X 1 3 3 1 1";
}
if (chordApplicature.Equals("A#sus4"))
{
    chordName = "X 1 3 3 4 1";
}
if (chordApplicature.Equals("A#m7"))
{
    chordName = "X 1 3 1 2 4";
}
if (chordApplicature.Equals("A#m7"))
{
```

```
    chordName = "X 1 3 1 2 1";
}
if (chordApplicature.Equals("A#7"))
{
    chordName = "X 1 3 1 3 4";
}
if (chordApplicature.Equals("A#7"))
{
    chordName = "X 1 3 1 3 1";
}
if (chordApplicature.Equals("A#6"))
{
    chordName = "X 1 0 0 3 1";
}
if (chordApplicature.Equals("A#m(VI)"))
{
    chordName = "6 8 8 6 6 6";
}
if (chordApplicature.Equals("A#(VI)"))
{
    chordName = "6 8 8 7 6 6";
}
if (chordApplicature.Equals("A#m(VI)"))
{
    chordName = "6 8 8 6 6 6";
}
if (chordApplicature.Equals("A#sus4(VI)"))
{
    chordName = "6 8 8 8 6 6";
}
if (chordApplicature.Equals("A#7(VI)"))
{
    chordName = "6 8 6 7 9 6";
}
if (chordApplicature.Equals("A#m7(VI)"))
{
    chordName = "6 8 6 6 9 6";
}
if (chordApplicature.Equals("A#6(III)"))
{
    chordName = "6 5 3 3 3 3";
}
if (chordApplicature.Equals("Bm"))
```

```
{
    chordName = "X 2 4 4 3 2";
}
if (chordApplicature.Equals("B"))
{
    chordName = "X 2 4 4 4 2";
}
if (chordApplicature.Equals("Bsus2"))
{
    chordName = "X 2 4 4 2 2";
}
if (chordApplicature.Equals("Bsus4"))
{
    chordName = "X 2 4 4 5 2";
}
if (chordApplicature.Equals("B7"))
{
    chordName = "X 2 4 2 4 2";
}
if (chordApplicature.Equals("B7"))
{
    chordName = "X 2 4 2 4 5";
}
if (chordApplicature.Equals("B7"))
{
    chordName = "X 2 1 2 0 2";
}
if (chordApplicature.Equals("B7sus4"))
{
    chordName = "X 2 4 2 5 2";
}
if (chordApplicature.Equals("B7sus4"))
{
    chordName = "X 2 4 2 5 5";
}
if (chordApplicature.Equals("B6"))
{
    chordName = "X 2 4 4 4 4";
}
if (chordApplicature.Equals("B(VII)"))
{
    chordName = "7 9 9 8 7 7";
}
```

```
if (chordApplicature.Equals("Bm(VII)"))
{
    chordName = "7 9 9 7 7 7";
}
if (chordApplicature.Equals("B7(VII)"))
{
    chordName = "7 9 7 8 10 7";
}
if (chordApplicature.Equals("Bm7(VII)"))
{
    chordName = "7 9 7 7 10 7";
}
if (chordApplicature.Equals("Bsus4(VII)"))
{
    chordName = "7 9 9 9 7 7";
}
if (chordApplicature.Equals("B6(IV)"))
{
    chordName = "7 6 4 4 4 4";
}
if (chordApplicature.Equals("C"))
{
    chordName = "X 3 2 0 1 0";
}
if (chordApplicature.Equals("C"))
{
    chordName = "X 3 5 5 5 3";
}
if (chordApplicature.Equals("Cm"))
{
    chordName = "X 3 5 5 4 3";
}
if (chordApplicature.Equals("Csus2"))
{
    chordName = "X 3 5 5 3 3";
}
if (chordApplicature.Equals("Csus4"))
{
    chordName = "X 3 5 5 6 3";
}
if (chordApplicature.Equals("C7"))
{
    chordName = "X 3 5 3 5 3";
```

```
}
if (chordApplicature.Equals("C7"))
{
    chordName = "X 3 5 3 5 6";
}
if (chordApplicature.Equals("C7sus4"))
{
    chordName = "X 3 5 3 6 3";
}
if (chordApplicature.Equals("Cm7"))
{
    chordName = "X 3 5 3 4 3";
}
if (chordApplicature.Equals("Cm7"))
{
    chordName = "X 3 5 3 4 6";
}
if (chordApplicature.Equals("C6"))
{
    chordName = "X 3 2 2 1 0";
}
if (chordApplicature.Equals("C(VIII)"))
{
    chordName = "8 10 10 9 8 8";
}
if (chordApplicature.Equals("Cm(VIII)"))
{
    chordName = "8 10 10 8 8 8";
}
if (chordApplicature.Equals("C7(VIII)"))
{
    chordName = "8 10 8 9 8 8";
}
if (chordApplicature.Equals("Cm7(VIII)"))
{
    chordName = "8 10 8 8 8 8";
}
if (chordApplicature.Equals("Csus4(VIII)"))
{
    chordName = "8 10 10 10 8 8";
}
if (chordApplicature.Equals("C#"))
{
```

```
    chordName = "X 4 6 6 6 4";
}
if (chordApplicature.Equals("C#m"))
{
    chordName = "X 4 6 6 5 4";
}
if (chordApplicature.Equals("C#sus2"))
{
    chordName = "X 4 6 6 4 4";
}
if (chordApplicature.Equals("C#sus4"))
{
    chordName = "X 4 6 6 7 4";
}
if (chordApplicature.Equals("C#7"))
{
    chordName = "X 4 6 4 6 4";
}
if (chordApplicature.Equals("C#7"))
{
    chordName = "X 4 6 4 6 7";
}
if (chordApplicature.Equals("C#m7"))
{
    chordName = "X 4 6 4 5 4";
}
if (chordApplicature.Equals("C#m7"))
{
    chordName = "X 4 6 4 5 7";
}
if (chordApplicature.Equals("C#7sus4"))
{
    chordName = "X 4 6 4 7 4";
}
if (chordApplicature.Equals("D"))
{
    chordName = "X X 0 2 3 2";
}
if (chordApplicature.Equals("D"))
{
    chordName = "X 0 0 2 3 2";
}
if (chordApplicature.Equals("D"))
```

```
{
    chordName = "X 5 7 7 7 5";
}
if (chordApplicature.Equals("Dm"))
{
    chordName = "X X 0 2 3 1";
}
if (chordApplicature.Equals("Dm"))
{
    chordName = "X 0 0 2 3 1";
}
if (chordApplicature.Equals("Dm"))
{
    chordName = "X 5 7 7 6 5";
}
if (chordApplicature.Equals("Dmaj7b5"))
{
    chordName = "X X 0 2 3 0";
}
if (chordApplicature.Equals("Dmaj7b5"))
{
    chordName = "X 0 0 2 3 0";
}
if (chordApplicature.Equals("Dsus2"))
{
    chordName = "X 5 7 7 5 5";
}
if (chordApplicature.Equals("Dsus4"))
{
    chordName = "X 5 7 7 8 5";
}
if (chordApplicature.Equals("D7"))
{
    chordName = "X 5 7 5 7 5";
}
if (chordApplicature.Equals("D7"))
{
    chordName = "X 5 7 5 7 8";
}
if (chordApplicature.Equals("Dm7"))
{
    chordName = "X 5 7 5 6 5";
}
```

```
if (chordApplicature.Equals("Dm7"))
{
    chordName = "X 5 7 5 6 8";
}
if (chordApplicature.Equals("D7sus4"))
{
    chordName = "X 5 7 5 8 5";
}
if (chordApplicature.Equals("D6"))
{
    chordName = "X X 0 2 0 2";
}
if (chordApplicature.Equals("D6"))
{
    chordName = "X 0 0 2 0 2";
}
if (chordApplicature.Equals("D#"))
{
    chordName = "X X 1 3 4 3";
}
if (chordApplicature.Equals("D#"))
{
    chordName = "X 6 8 8 8 6";
}
if (chordApplicature.Equals("D#m"))
{
    chordName = "X X 1 3 4 2";
}
if (chordApplicature.Equals("D#m"))
{
    chordName = "X 6 8 8 7 6";
}
if (chordApplicature.Equals("D#sus2"))
{
    chordName = "X 6 8 8 6 6";
}
if (chordApplicature.Equals("D#sus4"))
{
    chordName = "X 6 8 8 9 6";
}
if (chordApplicature.Equals("D#7"))
{
    chordName = "X 6 8 6 8 6";
```

```
}
if (chordApplicature.Equals("D#7"))
{
    chordName = "X 6 8 6 7 9";
}
if (chordApplicature.Equals("D#m7"))
{
    chordName = "X 6 8 6 7 6";
}
if (chordApplicature.Equals("D#m7"))
{
    chordName = "X 6 8 6 7 9";
}
if (chordApplicature.Equals("D#7sus4"))
{
    chordName = "X 6 8 6 9 6";
}
if (chordApplicature.Equals("E"))
{
    chordName = "0 2 2 1 0 0";
}
if (chordApplicature.Equals("Em"))
{
    chordName = "0 2 2 0 0 0";
}
if (chordApplicature.Equals("Esus4"))
{
    chordName = "0 2 2 2 0 0";
}
if (chordApplicature.Equals("E7"))
{
    chordName = "0 2 0 1 3 0";
}
if (chordApplicature.Equals("Em7"))
{
    chordName = "0 2 0 0 3 0";
}
if (chordApplicature.Equals("E6"))
{
    chordName = "0 2 2 1 1 0";
}
if (chordApplicature.Equals("Em6"))
{
```

```
    chordName = "0 2 2 0 1 0";
}
if (chordApplicature.Equals("Em9"))
{
    chordName = "0 2 2 0 0 2";
}
if (chordApplicature.Equals("Em9"))
{
    chordName = "0 2 2 0 0 3";
}
if (chordApplicature.Equals("E(VII)"))
{
    chordName = "X 7 9 9 9 7";
}
if (chordApplicature.Equals("Em(VII)"))
{
    chordName = "X 7 9 9 8 7";
}
if (chordApplicature.Equals("Esus2(VII)"))
{
    chordName = "X 7 9 9 7 7";
}
if (chordApplicature.Equals("Esus4(VII)"))
{
    chordName = "X 7 9 9 10 7";
}
if (chordApplicature.Equals("E7(VII)"))
{
    chordName = "X 7 9 7 9 7";
}
if (chordApplicature.Equals("E7(VII)"))
{
    chordName = "X 7 9 7 9 10";
}
if (chordApplicature.Equals("Em7(VII)"))
{
    chordName = "X 7 9 7 8 7";
}
if (chordApplicature.Equals("Em7(VII)"))
{
    chordName = "X 7 9 7 8 10";
}
if (chordApplicature.Equals("E7sus4(VII)"))
```

```
{
    chordName = "X 7 9 7 10 7";
}
if (chordApplicature.Equals("F"))
{
    chordName = "1 3 3 2 1 1";
}
if (chordApplicature.Equals("Fmaj"))
{
    chordName = "X 3 3 2 1 0";
}
if (chordApplicature.Equals("Fm"))
{
    chordName = "1 3 3 1 1 1";
}
if (chordApplicature.Equals("Fsus4"))
{
    chordName = "1 3 3 3 1 1";
}
if (chordApplicature.Equals("F7"))
{
    chordName = "1 3 1 2 4 1";
}
if (chordApplicature.Equals("F7"))
{
    chordName = "1 3 1 2 1 1";
}
if (chordApplicature.Equals("Fm7"))
{
    chordName = "1 3 1 1 1 1";
}
if (chordApplicature.Equals("Fm7"))
{
    chordName = "1 3 1 1 4 1";
}
if (chordApplicature.Equals("F6"))
{
    chordName = "1 0 0 2 1 0";
}
if (chordApplicature.Equals("F(VIII)"))
{
    chordName = "X 8 10 10 10 8";
}
```

```
if (chordApplicature.Equals("Fm(VIII)"))
{
    chordName = "X 8 10 10 9 8";
}
if (chordApplicature.Equals("F7(VIII)"))
{
    chordName = "X 8 10 8 10 8";
}
if (chordApplicature.Equals("F#"))
{
    chordName = "2 4 4 3 2 2";
}
if (chordApplicature.Equals("F#m"))
{
    chordName = "2 4 4 2 2 2";
}
if (chordApplicature.Equals("F#sus4"))
{
    chordName = "2 4 4 4 2 2";
}
if (chordApplicature.Equals("F#7"))
{
    chordName = "2 4 2 3 2 2";
}
if (chordApplicature.Equals("F#7"))
{
    chordName = "2 4 2 3 5 2";
}
if (chordApplicature.Equals("F#m7"))
{
    chordName = "2 4 2 2 2 2";
}
if (chordApplicature.Equals("F#m7"))
{
    chordName = "2 4 2 2 5 2";
}
if (chordApplicature.Equals("F#6"))
{
    chordName = "2 X 1 3 2 X";
}
if (chordApplicature.Equals("G"))
{
    chordName = "3 2 0 0 0 3";
```

```
}
if (chordApplicature.Equals("G"))
{
    chordName = "3 2 0 0 3 3";
}
if (chordApplicature.Equals("G(III)"))
{
    chordName = "3 5 5 4 3 3";
}
if (chordApplicature.Equals("Gm"))
{
    chordName = "3 5 5 3 3 3";
}
if (chordApplicature.Equals("Gsus4"))
{
    chordName = "3 5 5 5 3 3";
}
if (chordApplicature.Equals("G7"))
{
    chordName = "3 5 3 4 3 3";
}
if (chordApplicature.Equals("G7"))
{
    chordName = "3 5 3 4 6 3";
}
if (chordApplicature.Equals("Gm7"))
{
    chordName = "3 5 3 3 3 3";
}
if (chordApplicature.Equals("Gm7"))
{
    chordName = "3 5 3 3 6 3";
}
if (chordApplicature.Equals("G6"))
{
    chordName = "3 2 0 0 0 0";
}
if (chordApplicature.Equals("G#"))
{
    chordName = "4 6 6 5 4 4";
}
if (chordApplicature.Equals("G#m"))
{
```

```
            chordName = "4 6 6 4 4 4";
        }
        if (chordApplicature.Equals("G#sus4"))
        {
            chordName = "4 6 6 6 4 4";
        }
        if (chordApplicature.Equals("G#7"))
        {
            chordName = "4 6 4 5 4 4";
        }
        if (chordApplicature.Equals("G#7"))
        {
            chordName = "4 6 4 5 7 4";
        }
        if (chordApplicature.Equals("G#m7"))
        {
            chordName = "4 6 4 4 4 4";
        }
        if (chordApplicature.Equals("G#m7"))
        {
            chordName = "4 6 4 4 7 4";
        }

        textBox4.Text = chordName;
    }

    public void Decoder()
    {
        chordName = "No chord like this in database";
        if (chordApplicature.Equals("X 0 2 2 1 0"))
        {
            chordName = "Am";
        }
        if (chordApplicature.Equals("X 0 2 2 2 0"))
        {
            chordName = "A";
        }
        if (chordApplicature.Equals("X 0 2 2 0 0"))
        {
            chordName = "Asus2";
        }
        if (chordApplicature.Equals("X 0 2 2 3 0"))
        {
```

```
    chordName = "Asus4";
}
if (chordApplicature.Equals("X 0 2 0 1 0"))
{
    chordName = "Am7";
}
if (chordApplicature.Equals("X 0 2 0 1 3"))
{
    chordName = "Am7";
}
if (chordApplicature.Equals("X 0 2 0 2 0"))
{
    chordName = "A7";
}
if (chordApplicature.Equals("X 0 2 0 2 3"))
{
    chordName = "A7";
}
if (chordApplicature.Equals("X 0 2 2 2 2"))
{
    chordName = "A6";
}
if (chordApplicature.Equals("X 3 2 2 1 0"))
{
    chordName = "Am/C";
}
if (chordApplicature.Equals("5 7 7 5 5 5"))
{
    chordName = "Am(V)";
}
if (chordApplicature.Equals("5 7 7 6 5 5"))
{
    chordName = "A(V)";
}
if (chordApplicature.Equals("5 7 5 5 8 5"))
{
    chordName = "Am7(V)";
}
if (chordApplicature.Equals("5 7 7 7 5 5"))
{
    chordName = "Asus4(V)";
}
if (chordApplicature.Equals("5 7 5 6 8 5"))
```

```
{
    chordName = "A7(V)";
}
if (chordApplicature.Equals("5 7 7 5 7 5"))
{
    chordName = "Am6(V)";
}
if (chordApplicature.Equals("2 4 2 2 2 2"))
{
    chordName = "A6(V)";
}
if (chordApplicature.Equals("X 1 3 3 2 1"))
{
    chordName = "A#m";
}
if (chordApplicature.Equals("X 1 3 3 3 1"))
{
    chordName = "A#";
}
if (chordApplicature.Equals("X 1 3 3 1 1"))
{
    chordName = "A#sus2";
}
if (chordApplicature.Equals("X 1 3 3 4 1"))
{
    chordName = "A#sus4";
}
if (chordApplicature.Equals("X 1 3 1 2 4"))
{
    chordName = "A#m7";
}
if (chordApplicature.Equals("X 1 3 1 2 1"))
{
    chordName = "A#m7";
}
if (chordApplicature.Equals("X 1 3 1 3 4"))
{
    chordName = "A#7";
}
if (chordApplicature.Equals("X 1 3 1 3 1"))
{
    chordName = "A#7";
}
```

```
if (chordApplicature.Equals("X 1 0 0 3 1"))
{
    chordName = "A#6";
}
if (chordApplicature.Equals("6 8 8 6 6 6"))
{
    chordName = "A#m(VI)";
}
if (chordApplicature.Equals("6 8 8 7 6 6"))
{
    chordName = "A#(VI)";
}
if (chordApplicature.Equals("6 8 8 6 6 6"))
{
    chordName = "A#m(VI)";
}
if (chordApplicature.Equals("6 8 8 8 6 6"))
{
    chordName = "A#sus4(VI)";
}
if (chordApplicature.Equals("6 8 6 7 9 6"))
{
    chordName = "A#7(VI)";
}
if (chordApplicature.Equals("6 8 6 6 9 6"))
{
    chordName = "A#m7(VI)";
}
if (chordApplicature.Equals("6 5 3 3 3 3"))
{
    chordName = "A#6(III)";
}
if (chordApplicature.Equals("X 2 4 4 3 2"))
{
    chordName = "Bm";
}
if (chordApplicature.Equals("X 2 4 4 4 2"))
{
    chordName = "B";
}
if (chordApplicature.Equals("X 2 4 4 2 2"))
{
    chordName = "Bsus2";
```

```
}
if (chordApplicature.Equals("X 2 4 4 5 2"))
{
    chordName = "Bsus4";
}
if (chordApplicature.Equals("X 2 4 2 4 2"))
{
    chordName = "B7";
}
if (chordApplicature.Equals("X 2 4 2 4 5"))
{
    chordName = "B7";
}
if (chordApplicature.Equals("X 2 1 2 0 2"))
{
    chordName = "B7";
}
if (chordApplicature.Equals("X 2 4 2 5 2"))
{
    chordName = "B7sus4";
}
if (chordApplicature.Equals("X 2 4 2 5 5"))
{
    chordName = "B7sus4";
}
if (chordApplicature.Equals("X 2 4 4 4 4"))
{
    chordName = "B6";
}
if (chordApplicature.Equals("7 9 9 8 7 7"))
{
    chordName = "B(VII)";
}
if (chordApplicature.Equals("7 9 9 7 7 7"))
{
    chordName = "Bm(VII)";
}
if (chordApplicature.Equals("7 9 7 8 10 7"))
{
    chordName = "B7(VII)";
}
if (chordApplicature.Equals("7 9 7 7 10 7"))
{
```

```
    chordName = "Bm7(VII)";
}
if (chordApplicature.Equals("7 9 9 9 7 7"))
{
    chordName = "Bsus4(VII)";
}
if (chordApplicature.Equals("7 6 4 4 4 4"))
{
    chordName = "B6(IV)";
}
if (chordApplicature.Equals("X 3 2 0 1 0"))
{
    chordName = "C";
}
if (chordApplicature.Equals("X 3 5 5 5 3"))
{
    chordName = "C";
}
if (chordApplicature.Equals("X 3 5 5 4 3"))
{
    chordName = "Cm";
}
if (chordApplicature.Equals("X 3 5 5 3 3"))
{
    chordName = "Csus2";
}
if (chordApplicature.Equals("X 3 5 5 6 3"))
{
    chordName = "Csus4";
}
if (chordApplicature.Equals("X 3 5 3 5 3"))
{
    chordName = "C7";
}
if (chordApplicature.Equals("X 3 5 3 5 6"))
{
    chordName = "C7";
}
if (chordApplicature.Equals("X 3 5 3 6 3"))
{
    chordName = "C7sus4";
}
if (chordApplicature.Equals("X 3 5 3 4 3"))
```

```
{
    chordName = "Cm7";
}
if (chordApplicature.Equals("X 3 5 3 4 6"))
{
    chordName = "Cm7";
}
if (chordApplicature.Equals("X 3 2 2 1 0"))
{
    chordName = "C6";
}
if (chordApplicature.Equals("8 10 10 9 8 8"))
{
    chordName = "C(VIII)";
}
if (chordApplicature.Equals("8 10 10 8 8 8"))
{
    chordName = "Cm(VIII)";
}
if (chordApplicature.Equals("8 10 8 9 8 8"))
{
    chordName = "C7(VIII)";
}
if (chordApplicature.Equals("8 10 8 8 8 8"))
{
    chordName = "Cm7(VIII)";
}
if (chordApplicature.Equals("8 10 10 10 8 8"))
{
    chordName = "Csus4(VIII)";
}
if (chordApplicature.Equals("X 4 6 6 6 4"))
{
    chordName = "C#";
}
if (chordApplicature.Equals("X 4 6 6 5 4"))
{
    chordName = "C#m";
}
if (chordApplicature.Equals("X 4 6 6 4 4"))
{
    chordName = "C#sus2";
}
```

```
if (chordApplicature.Equals("X 4 6 6 7 4"))
{
    chordName = "C#sus4";
}
if (chordApplicature.Equals("X 4 6 4 6 4"))
{
    chordName = "C#7";
}
if (chordApplicature.Equals("X 4 6 4 6 7"))
{
    chordName = "C#7";
}
if (chordApplicature.Equals("X 4 6 4 5 4"))
{
    chordName = "C#m7";
}
if (chordApplicature.Equals("X 4 6 4 5 7"))
{
    chordName = "C#m7";
}
if (chordApplicature.Equals("X 4 6 4 7 4"))
{
    chordName = "C#7sus4";
}
if (chordApplicature.Equals("X X 0 2 3 2"))
{
    chordName = "D";
}
if (chordApplicature.Equals("X 0 0 2 3 2"))
{
    chordName = "D";
}
if (chordApplicature.Equals("X 5 7 7 7 5"))
{
    chordName = "D";
}
if (chordApplicature.Equals("X X 0 2 3 1"))
{
    chordName = "Dm";
}
if (chordApplicature.Equals("X 0 0 2 3 1"))
{
    chordName = "Dm";
```

```
}
if (chordApplicature.Equals("X 5 7 7 6 5"))
{
    chordName = "Dm";
}
if (chordApplicature.Equals("X X 0 2 3 0"))
{
    chordName = "Dmaj7b5";
}
if (chordApplicature.Equals("X 0 0 2 3 0"))
{
    chordName = "Dmaj7b5";
}
if (chordApplicature.Equals("X 5 7 7 5 5"))
{
    chordName = "Dsus2";
}
if (chordApplicature.Equals("X 5 7 7 8 5"))
{
    chordName = "Dsus4";
}
if (chordApplicature.Equals("X 5 7 5 7 5"))
{
    chordName = "D7";
}
if (chordApplicature.Equals("X 5 7 5 7 8"))
{
    chordName = "D7";
}
if (chordApplicature.Equals("X 5 7 5 6 5"))
{
    chordName = "Dm7";
}
if (chordApplicature.Equals("X 5 7 5 6 8"))
{
    chordName = "Dm7";
}
if (chordApplicature.Equals("X 5 7 5 8 5"))
{
    chordName = "D7sus4";
}
if (chordApplicature.Equals("X X 0 2 0 2"))
{
```

```
    chordName = "D6";
}
if (chordApplicature.Equals("X 0 0 2 0 2"))
{
    chordName = "D6";
}
if (chordApplicature.Equals("X X 1 3 4 3"))
{
    chordName = "D#";
}
if (chordApplicature.Equals("X 6 8 8 8 6"))
{
    chordName = "D#";
}
if (chordApplicature.Equals("X X 1 3 4 2"))
{
    chordName = "D#m";
}
if (chordApplicature.Equals("X 6 8 8 7 6"))
{
    chordName = "D#m";
}
if (chordApplicature.Equals("X 6 8 8 6 6"))
{
    chordName = "D#sus2";
}
if (chordApplicature.Equals("X 6 8 8 9 6"))
{
    chordName = "D#sus4";
}
if (chordApplicature.Equals("X 6 8 6 8 6"))
{
    chordName = "D#7";
}
if (chordApplicature.Equals("X 6 8 6 7 9"))
{
    chordName = "D#7";
}
if (chordApplicature.Equals("X 6 8 6 7 6"))
{
    chordName = "D#m7";
}
if (chordApplicature.Equals("X 6 8 6 7 9"))
```

```
{
    chordName = "D#m7";
}
if (chordApplicature.Equals("X 6 8 6 9 6"))
{
    chordName = "D#7sus4";
}
if (chordApplicature.Equals("0 2 2 1 0 0"))
{
    chordName = "E";
}
if (chordApplicature.Equals("0 2 2 0 0 0"))
{
    chordName = "Em";
}
if (chordApplicature.Equals("0 2 2 2 0 0"))
{
    chordName = "Esus4";
}
if (chordApplicature.Equals("0 2 0 1 3 0"))
{
    chordName = "E7";
}
if (chordApplicature.Equals("0 2 0 0 3 0"))
{
    chordName = "Em7";
}
if (chordApplicature.Equals("0 2 2 1 1 0"))
{
    chordName = "E6";
}
if (chordApplicature.Equals("0 2 2 0 1 0"))
{
    chordName = "Em6";
}
if (chordApplicature.Equals("0 2 2 0 0 2"))
{
    chordName = "Em9";
}
if (chordApplicature.Equals("0 2 2 0 0 3"))
{
    chordName = "Em9";
}
```

```
if (chordApplicature.Equals("X 7 9 9 9 7"))
{
    chordName = "E(VII)";
}
if (chordApplicature.Equals("X 7 9 9 8 7"))
{
    chordName = "Em(VII)";
}
if (chordApplicature.Equals("X 7 9 9 7 7"))
{
    chordName = "Esus2(VII)";
}
if (chordApplicature.Equals("X 7 9 9 10 7"))
{
    chordName = "Esus4(VII)";
}
if (chordApplicature.Equals("X 7 9 7 9 7"))
{
    chordName = "E7(VII)";
}
if (chordApplicature.Equals("X 7 9 7 9 10"))
{
    chordName = "E7(VII)";
}
if (chordApplicature.Equals("X 7 9 7 8 7"))
{
    chordName = "Em7(VII)";
}
if (chordApplicature.Equals("X 7 9 7 8 10"))
{
    chordName = "Em7(VII)";
}
if (chordApplicature.Equals("X 7 9 7 10 7"))
{
    chordName = "E7sus4(VII)";
}
if (chordApplicature.Equals("1 3 3 2 1 1"))
{
    chordName = "F";
}
if (chordApplicature.Equals("X 3 3 2 1 0"))
{
    chordName = "Fmaj";
```

```
}
if (chordApplicature.Equals("1 3 3 1 1 1"))
{
    chordName = "Fm";
}
if (chordApplicature.Equals("1 3 3 3 1 1"))
{
    chordName = "Fsus4";
}
if (chordApplicature.Equals("1 3 1 2 4 1"))
{
    chordName = "F7";
}
if (chordApplicature.Equals("1 3 1 2 1 1"))
{
    chordName = "F7";
}
if (chordApplicature.Equals("1 3 1 1 1 1"))
{
    chordName = "Fm7";
}
if (chordApplicature.Equals("1 3 1 1 4 1"))
{
    chordName = "Fm7";
}
if (chordApplicature.Equals("1 0 0 2 1 0"))
{
    chordName = "F6";
}
if (chordApplicature.Equals("X 8 10 10 10 8"))
{
    chordName = "F(VIII)";
}
if (chordApplicature.Equals("X 8 10 10 9 8"))
{
    chordName = "Fm(VIII)";
}
if (chordApplicature.Equals("X 8 10 8 10 8"))
{
    chordName = "F7(VIII)";
}
if (chordApplicature.Equals("2 4 4 3 2 2"))
{
```

```
    chordName = "F#";
}
if (chordApplicature.Equals("2 4 4 2 2 2"))
{
    chordName = "F#m";
}
if (chordApplicature.Equals("2 4 4 4 2 2"))
{
    chordName = "F#sus4";
}
if (chordApplicature.Equals("2 4 2 3 2 2"))
{
    chordName = "F#7";
}
if (chordApplicature.Equals("2 4 2 3 5 2"))
{
    chordName = "F#7";
}
if (chordApplicature.Equals("2 4 2 2 2 2"))
{
    chordName = "F#m7";
}
if (chordApplicature.Equals("2 4 2 2 5 2"))
{
    chordName = "F#m7";
}
if (chordApplicature.Equals("2 X 1 3 2 X"))
{
    chordName = "F#6";
}
if (chordApplicature.Equals("3 2 0 0 0 3"))
{
    chordName = "G";
}
if (chordApplicature.Equals("3 2 0 0 3 3"))
{
    chordName = "G";
}
if (chordApplicature.Equals("3 5 5 4 3 3"))
{
    chordName = "G(III)";
}
if (chordApplicature.Equals("3 5 5 3 3 3"))
```

```
{
    chordName = "Gm";
}
if (chordApplicature.Equals("3 5 5 5 3 3"))
{
    chordName = "Gsus4";
}
if (chordApplicature.Equals("3 5 3 4 3 3"))
{
    chordName = "G7";
}
if (chordApplicature.Equals("3 5 3 4 6 3"))
{
    chordName = "G7";
}
if (chordApplicature.Equals("3 5 3 3 3 3"))
{
    chordName = "Gm7";
}
if (chordApplicature.Equals("3 5 3 3 6 3"))
{
    chordName = "Gm7";
}
if (chordApplicature.Equals("3 2 0 0 0 0"))
{
    chordName = "G6";
}
if (chordApplicature.Equals("4 6 6 5 4 4"))
{
    chordName = "G#";
}
if (chordApplicature.Equals("4 6 6 4 4 4"))
{
    chordName = "G#m";
}
if (chordApplicature.Equals("4 6 6 6 4 4"))
{
    chordName = "G#sus4";
}
if (chordApplicature.Equals("4 6 4 5 4 4"))
{
    chordName = "G#7";
}
```

```
    if (chordApplicature.Equals("4 6 4 5 7 4"))
    {
        chordName = "G#7";
    }
    if (chordApplicature.Equals("4 6 4 4 4 4"))
    {
        chordName = "G#m7";
    }
    if (chordApplicature.Equals("4 6 4 4 7 4"))
    {
        chordName = "G#m7";
    }

    textBox2.Text = chordName;
}


public Form1()
{
    InitializeComponent();
}

private void pictureBox1_Click(object sender, EventArgs e)
{

}

private void checkBox1_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox34_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox41_CheckedChanged(object sender, EventArgs e)
{

}

private void button1_Click(object sender, EventArgs e)
```

```
{
    CheckFirstString();
    CheckSecondString();
    CheckThirdString();
    CheckFourthString();
    CheckFifthString();
    CheckSixthString();
    TransformFirstString();
    TransformSecondString();
    TransformThirdString();
    TransformFourthString();
    TransformFifthString();
    TransformSixthString();
    BuildChord();
    Decoder();
}

private void checkBox62_CheckedChanged(object sender, EventArgs e)
{

}

private void label1_Click(object sender, EventArgs e)
{

}

private void label5_Click(object sender, EventArgs e)
{

}

private void label11_Click(object sender, EventArgs e)
{

}

private void button2_Click(object sender, EventArgs e)
{
    CheckChordName();
    Coder();
}
```

```
private void button3_Click(object sender, EventArgs e)
{
    Hide();
    Quart_Quint_Circle form2 = new Quart_Quint_Circle();
    form2.ShowDialog();
    Close();
}

private void checkBox61_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox2_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox3_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox4_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox5_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox6_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox7_CheckedChanged(object sender, EventArgs e)
{

}
```

```
private void checkBox8_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox9_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox10_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox11_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox12_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox13_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox14_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox15_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox16_CheckedChanged(object sender, EventArgs e)
{
```

```
}

private void checkBox17_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox18_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox19_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox20_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox63_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox21_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox22_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox23_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox24_CheckedChanged(object sender, EventArgs e)
```

```
{

}

private void checkBox25_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox26_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox27_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox28_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox29_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox30_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox65_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox31_CheckedChanged(object sender, EventArgs e)
{

}
```

```
private void checkBox32_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox33_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox35_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox36_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox37_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox38_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox39_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox40_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox42_CheckedChanged(object sender, EventArgs e)
{
```

```
}

private void checkBox43_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox44_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox45_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox46_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox47_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox48_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox49_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox50_CheckedChanged(object sender, EventArgs e)
{

}
```

```
private void checkBox51_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox52_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox53_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox54_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox55_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox56_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox57_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox58_CheckedChanged(object sender, EventArgs e)
{

}

private void checkBox59_CheckedChanged(object sender, EventArgs e)
{
```

```
            }

            private void checkBox60_CheckedChanged(object sender, EventArgs e)
            {

            }

            private void checkBox66_CheckedChanged(object sender, EventArgs e)
            {

            }

            private void checkBox64_CheckedChanged(object sender, EventArgs e)
            {

            }
        }
    }
```

Quint-Quar-Circle.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DyplomProgram
{
    public partial class Quart_Quint_Circle : Form
    {
        public Quart_Quint_Circle()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Hide();
            Form1 fooorm = new Form1();
            fooorm.ShowDialog();
            Close();
        }

        private void Quart_Quint_Circle_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace DyplomProgram
{
    static class Program
    {

        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Додаток Г - ГРАФІЧНА ЧАСТИНА

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ НАВЧАЛЬНОЇ СИСТЕМИ ГРИ НА ГІТАРІ

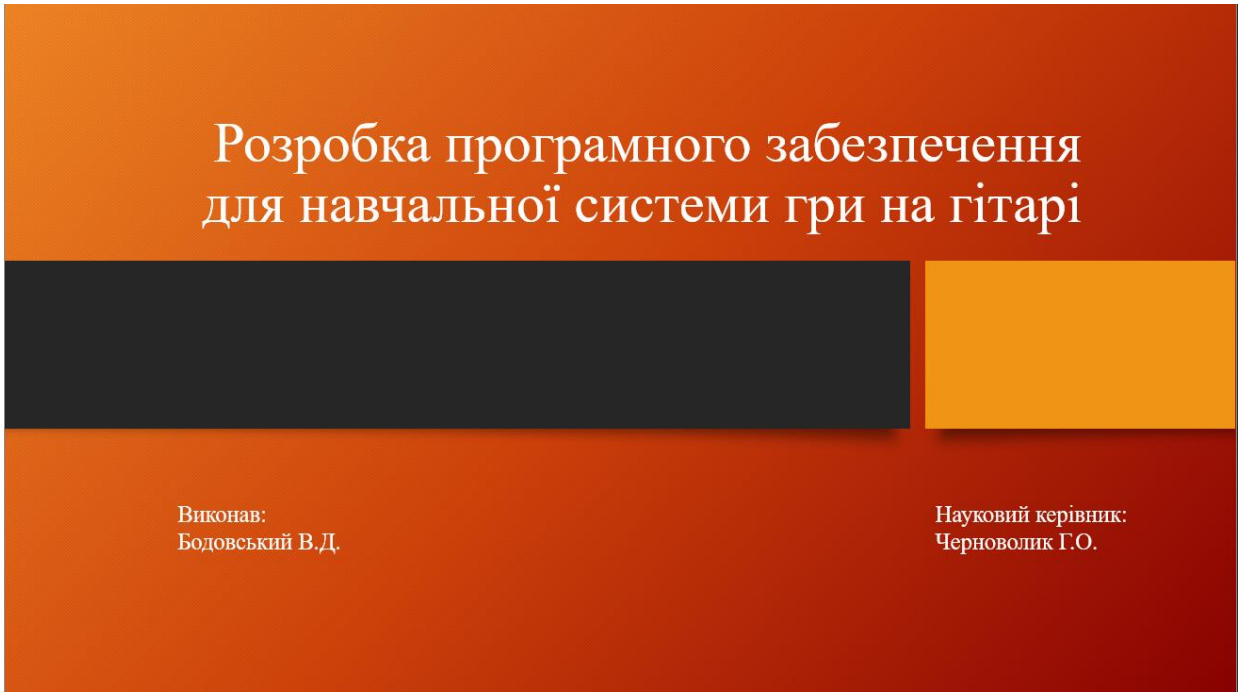

Рисунок Г.1 – Титульна сторінка презентації

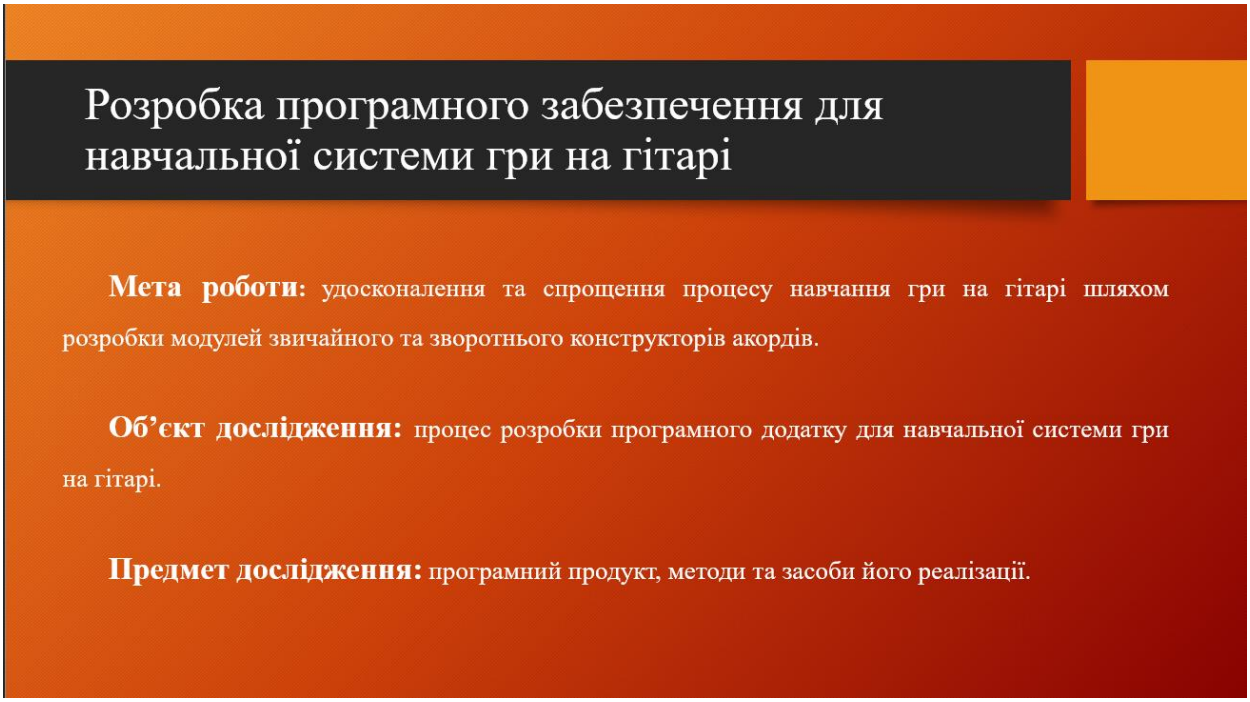

Рисунок Г.2 – Мета роботи, об`єкт дослідження, предмет дослідження

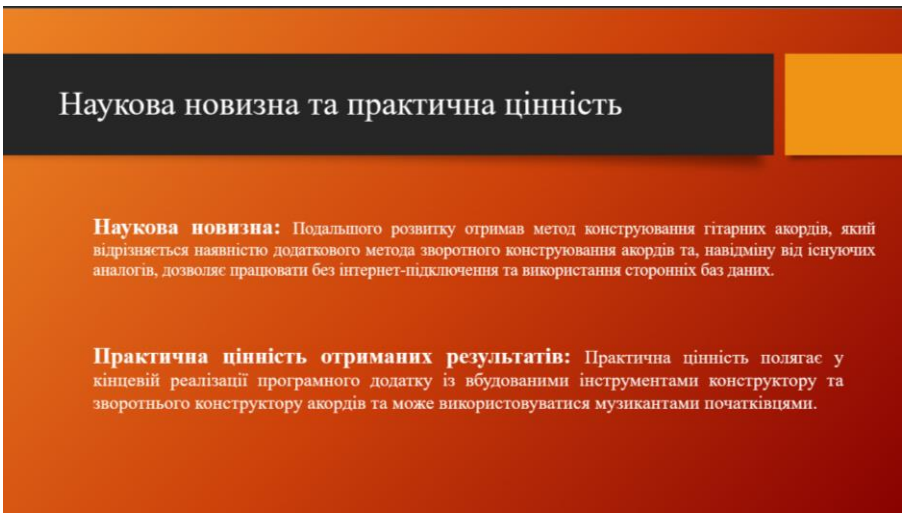

Рисунок Г.3 – Наукова новизна та практична цінність роботи

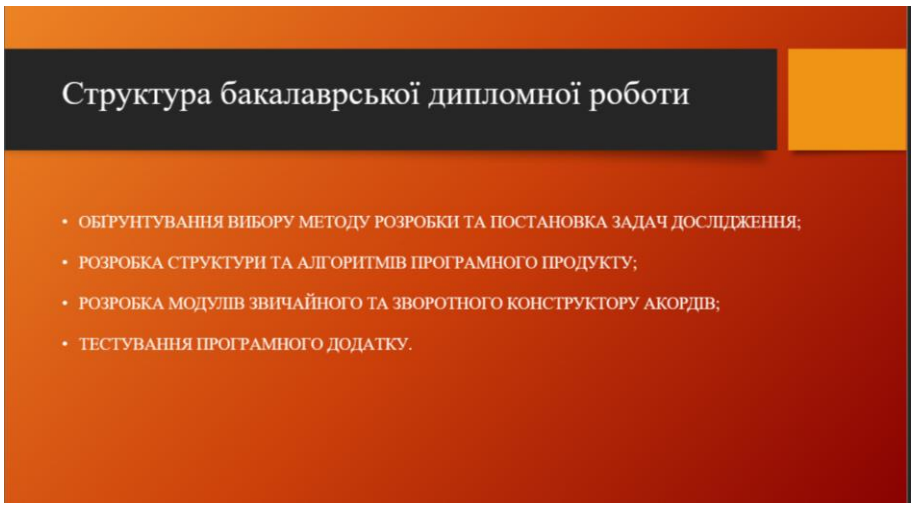

Рисунок Г.4 – Структура бакалаврської дипломної роботи

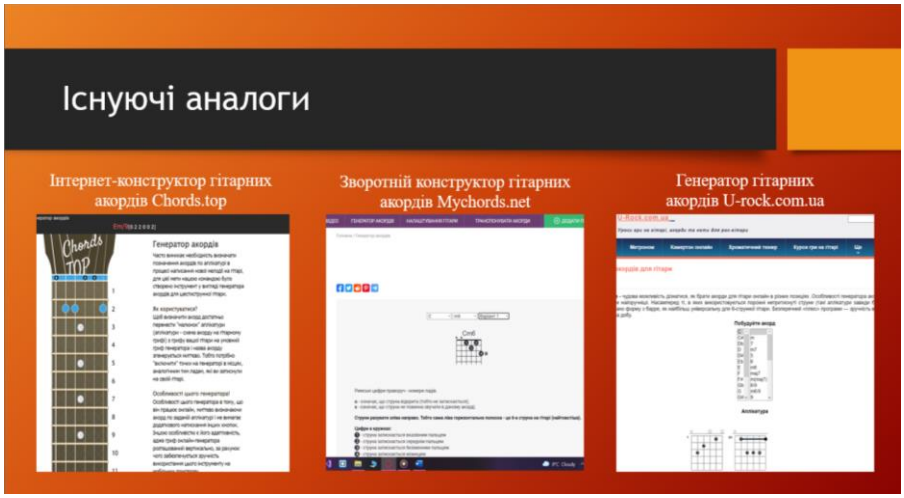

Рисунок Г.5 – Існуючі аналоги

Рисунок Г.6 - Структура інтерфейсу та вікон програми

Рисунок Г.7 - Алгоритм роботи головного вікна програми

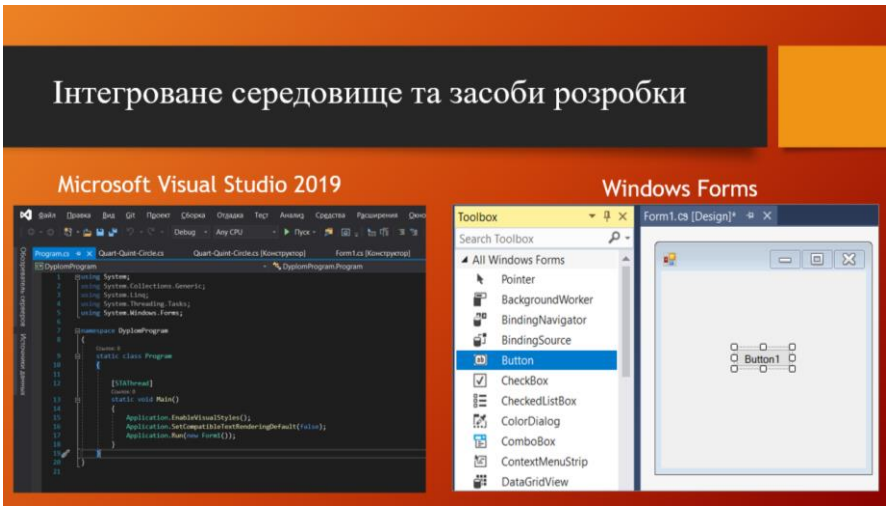

Рисунок Г.8 - Інтегроване середовище та засоби розробки

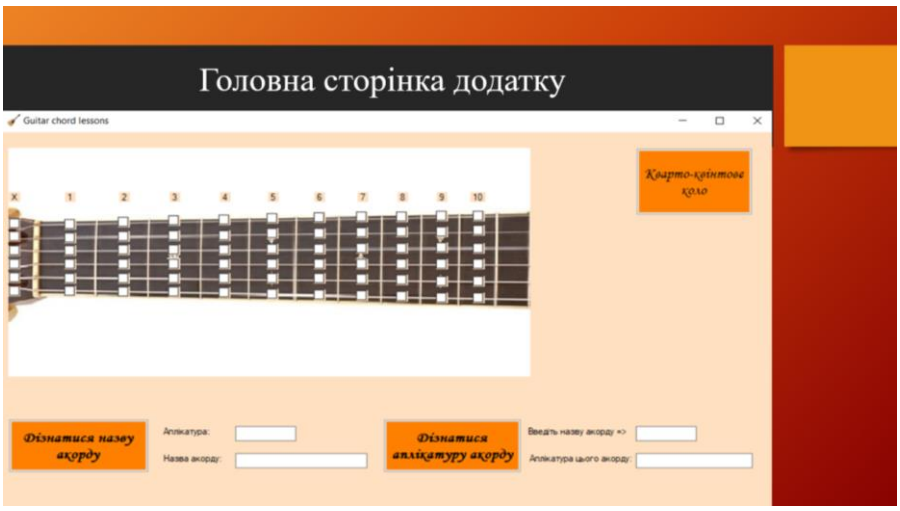

Рисунок Г.9 - Головна сторінка додатку

Рисунок Г.10 - Конструктор гітарних акордів

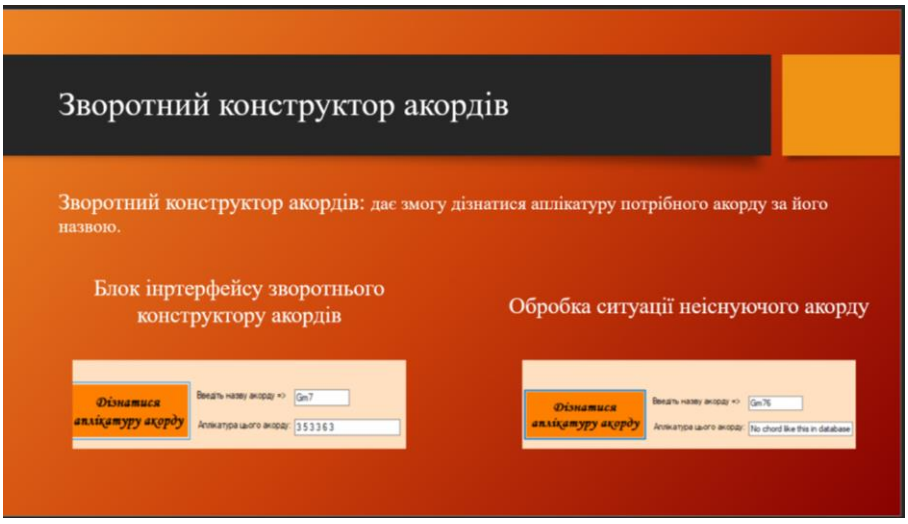

Рисунок Г.11 - Зворотний конструктор акордів

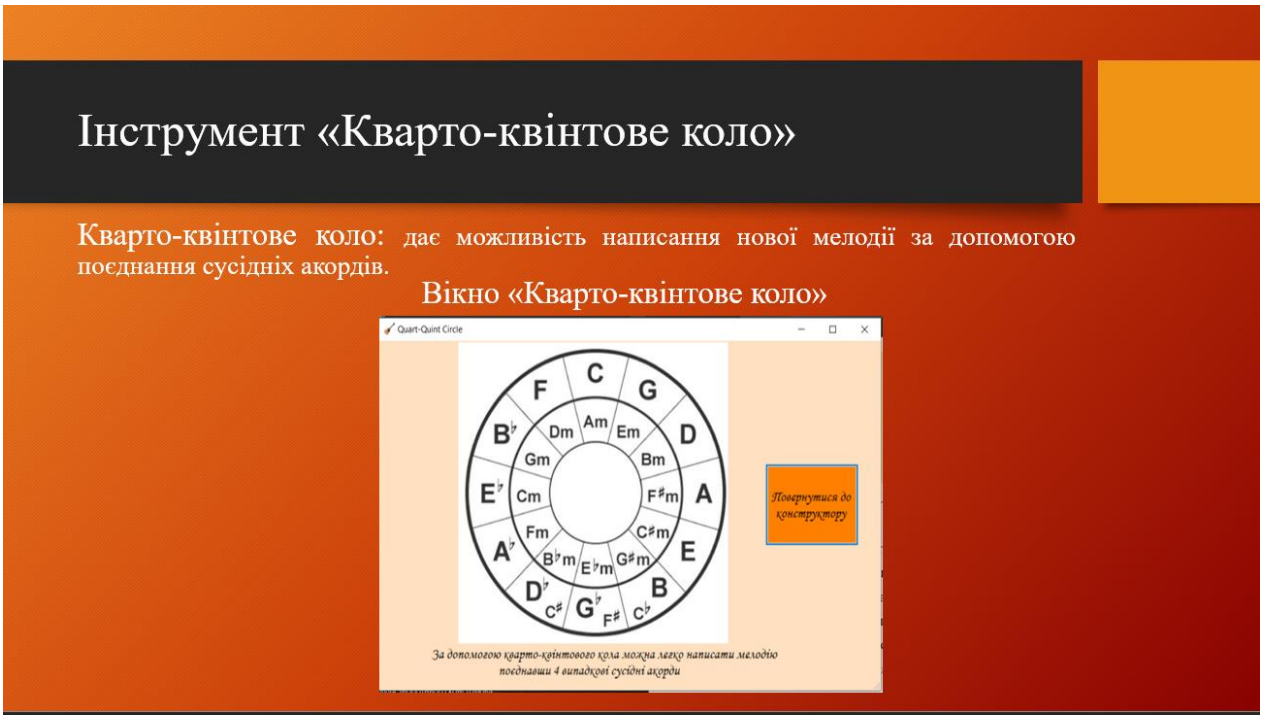

Рисунок Г.12 - Інструмент «Кварто-квінтове коло»

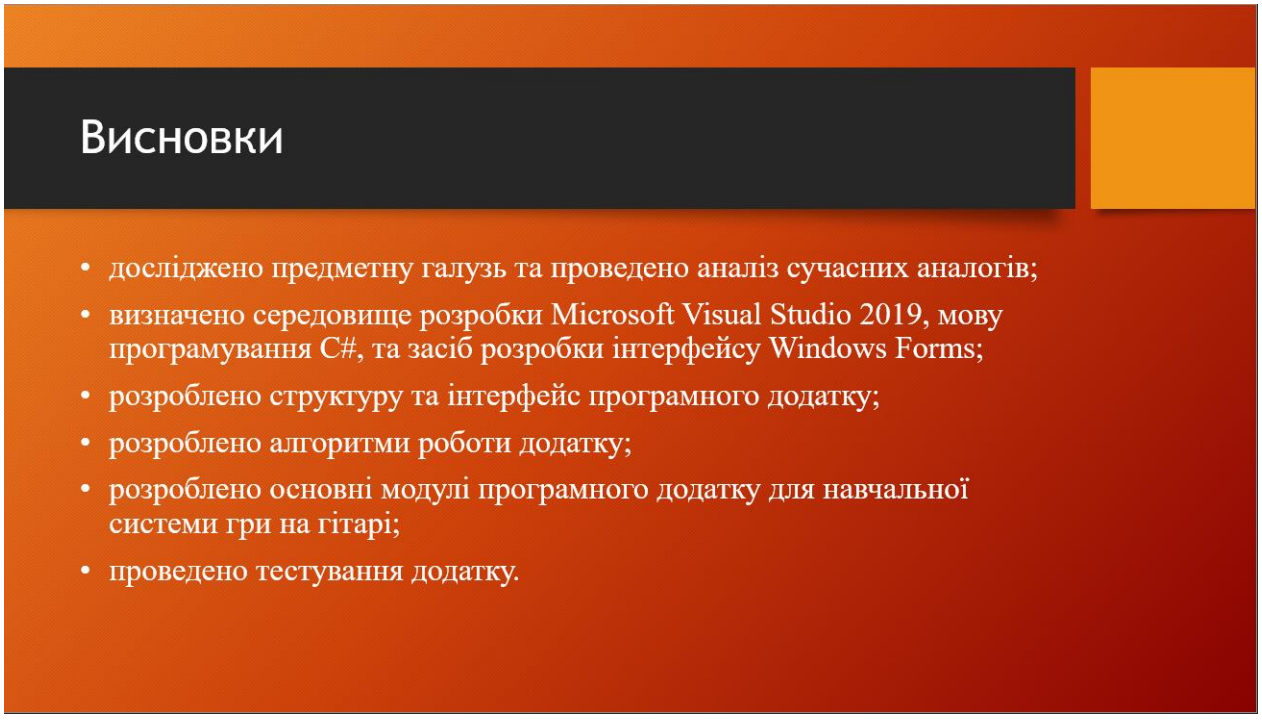

Рисунок Г.13 - Висновки