Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ «Информатика и системы управления»

КАФЕДРА ИУ-7 «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод генерации музыкального фрагмента, соответствующего эмоциональному состоянию человека, с использованием марковских моделей»

Студент группы ИУ7-81Б _____ Е.В. Фролова
(Подпись, дата) (И.О.Фамилия)

Руководитель ВКР _____ Ю.М. Гаврилова
(Подпись, дата) (И.О.Фамилия)

Нормоконтролер _____ _____
(Подпись, дата) (И.О.Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка содержит 81 страниц, 36 рисунков, 8 таблицы, 27 источников, 1 приложение.

Ключевые слова: MIDI, алгоритмическая композиция, марковские модели, схемы соотнесения цвета и ноты.

Цель бакалаврской работы – разработка метода генерации музыкального фрагмента, соответствующего эмоциональному состоянию человека, с использованием марковских моделей.

Задачи, решаемые в работе:

1. Анализ предметной области генерации музыкальной композиции и существующих методов алгоритмической композиции.
2. Разработка метода генерации музыкального фрагмента, соответсвующего эмоциональному состоянию человека, с использованием марковских моделей.
3. Разработка программного обеспечения, реализующего спроектированный метод.
4. Проведение исследования соответствия сгенерированных фрагментов заранее сформированной оценке методом экспертной оценки.

Разработанный метод может использоваться для генерация фоновой музыки для учебы/занятий спортом или для использования в коммерции.

В данной выпускной бакалаврской работе проанализированы существующие методы алгоритмической композици, проведен их сравнительный анализ и выбран наиболее подходящий для решаемой задачи метод. Разработан метод генерации музыкального фрагмента, соответсвующего эмоциональному состоянию человека и ПО, реализующее описанный метод. Проведено исследование соответствия сгенерированных фрагментов заранее сформированной оценке методом экспертной оценки.

СОДЕРЖАНИЕ

Введение

В настоящее время публикуется множество работ, направленных на автоматическую генерацию музыкальных композиций с использованием различных методов и подходов. Алгоритмическая композиция позволяет создавать уникальную музыку, не похожую на существующие произведения, что может помочь удовлетворить спрос на новые и оригинальные композиции. Также алгоритмическая композиция может ускорить процесс создания музыки и сделать его более эффективным. К тому же генеративная музыка – это тот тип музыки, который максимально гармонично сочетается с другой деятельностью, и как следствие, отлично подходит для учебы, работы, медитаций, занятий спортом и т. д.

Целью данной работы является разработка метода генерации музыкального фрагмента, соответствующего эмоциональному состоянию человека, с использованием марковских моделей.

В рамках работы требуется решить следующие **задачи**:

1. Провести анализ предметной области генерации музыкальной композиции и существующих методов алгоритмической композиции.
2. Разработать метод генерации музыкального фрагмента, соответствующего эмоциональному состоянию человека, с использованием марковских моделей.
3. Разработать программное обеспечение, реализующее спроектированный метод.
4. Провести исследование соответствия сгенерированных фрагментов заранее сформированной оценке методом экспертной оценки.

1 Аналитический раздел

1.1 Анализ предметной области

1.1.1 Актуальность задачи

Алгоритмическая композиция – это процесс создания музыки с помощью алгоритмов и компьютерных программ. Она позволяет создавать уникальную музыку, которая не похожа на другие произведения, а также может ускорить процесс создания музыки и сделать его более эффективным.

Как пример актуальности генеративной музыки можно привести Mubert[1], сервис для создания генеративной музыки. Алгоритм берёт семплы из библиотеки звуков и создаёт бесконечный музыкальный поток. Пользователь может выбирать жанр и темп музыки. Данный сервис пользуется большой популярностью, которую подтверждают подписки и прослушивания, происходящие в Mubert каждый день.

Главным недостатком большинства существующих программ для генерации музыкального трека является то, что музыка, которую они генерируют, лишена смысловой нагрузки – компьютеры не имеют чувств и настроений, не вкладывают в музыку переживания, свойственные человеку. Именно поэтому в данной работе генерация музыкального фрагмента будет основана на эмоциональном состоянии человека.

Генерация музыки, соответствующей эмоциональному состоянию человека, может помочь ему находить музыку, которая поднимает настроение, помогает расслабиться, а также положительно влиять на физиологические процессы, такие как сердечный ритм и дыхание. Это особенно актуально в условиях стресса и тревоги, когда человеку нужна поддержка и успокоение. Также эта тема акутальна для использования в медицине и психотерапии. Генерация музыки, соответствующей эмоциональному состоянию пациента, может повысить эффективность музыкальной терапии и улучшить результаты лечения различных психических расстройств, таких как депрессия,

тревожность и т. д. Она может помочь пациентам расслабиться, уменьшить уровень стресса и тревоги, а также улучшить настроение и самочувствие.

1.1.2 Существующие решения

Для определения способа решения поставленной задачи, необходимо проанализировать уже существующие на рынке решения и подходы, которые они используют для генерации мелодии.

На данный момент существует большое количество различных приложений для генерации музыки. Наиболее известными из них являются:

— Mubert [1] – сервис для создания генеративной музыки. Алгоритм использует нейронные сети, он берёт семплы из библиотеки звуков и создаёт бесконечный музыкальный поток.

— LMUSE [2] – алгоритм интерпретирует трехмерные Л-системы как MIDI файлы или сгенерированные MIDI звуки, затем Л-системы рекурсивно преобразуют строку символов в соответствии с заданным набором правил.

— BachBot [3] – это автоматическая система стилистической композиции для сочинения полифонической музыки в стиле хоралов Баха. Алгоритм использует модель на основе LSTM-нейросети.

— DeepJazz [4] – приложение для генерации джазовой музыки. Алгоритм использует двухслойную LSTM-нейросеть, которая обучается на предоставленных ей MIDI файлах.

— Magenta NSynth [5] – это нейронный синтезатор. В основе алгоритма лежит система искусственного интеллекта, которая миксует несколько предварительно загруженных сэмплов в новый звук с уникальными характеристиками.

— Computoser [6] – использует гибридный алгоритм для музыкальной композиции, основанный на вероятности и наборе правил.

— Meldy [7] – это простой генератор мелодий на основе предоставленного пользователем настроении. Алгоритм использует грамматическую модель.

Ниже приведена таблица существующих решений и методов, которые они используют для генерации музыкальных композиций.

Таблица 1.1 – Существующие решения

Решение	Используемый **метод**
Mubert	Искуственные нейронные сети
LMUSE	Л-системы
BachBot	Искуственные нейронные сети
DeepJazz	Искуственные нейронные сети
Magenta NSynth	Искуственные нейронные сети
Computoser	Вероятностная модель с использованием набора правил
Meldy	Грамматическая модель

Большая часть рассмотренных приложений использует алгоритмы, основанные на применении нейронных сетей, но также встречается использование алгоритмов, основанных на Л-системах и грамматических или вероятностных моделях.

1.2 Классфикация методов генерации музыкальной композиции

Алгоритмической музыкой (алгоритмической музыкальной композицией) называют процесс создания музыкальных отрывков, последовательностей и композиций с помощью математических моделей, правил и алгоритмов. Результатом алгоритмической музыки может стать создание оригинального произведения, сочинение в стиле определенной музыки, интерактивная

генерация произведения.

Существует два основных подхода к алгоритмической композиции. Первый метод основан на моделях, которые используют правила композиции, определенные человеком произвольным образом. Второй основан на моделях, в которых правила композиции генерируются на основе других музыкальных произведений, являющихся эталонными.

Все методы алгоритмической композиции можно разделить на следующие группы [8]:

1. Грамматические методы.
2. Математические модели (вероятностные методы).
3. Эволюционные методы (генетические алгоритмы, муравьиный алгоритм, Л-системы и клеточные автоматы).
4. Обучающиеся системы.

Ниже, на рисунке 1.1, представлена классификация методов алгоритмической композиции.

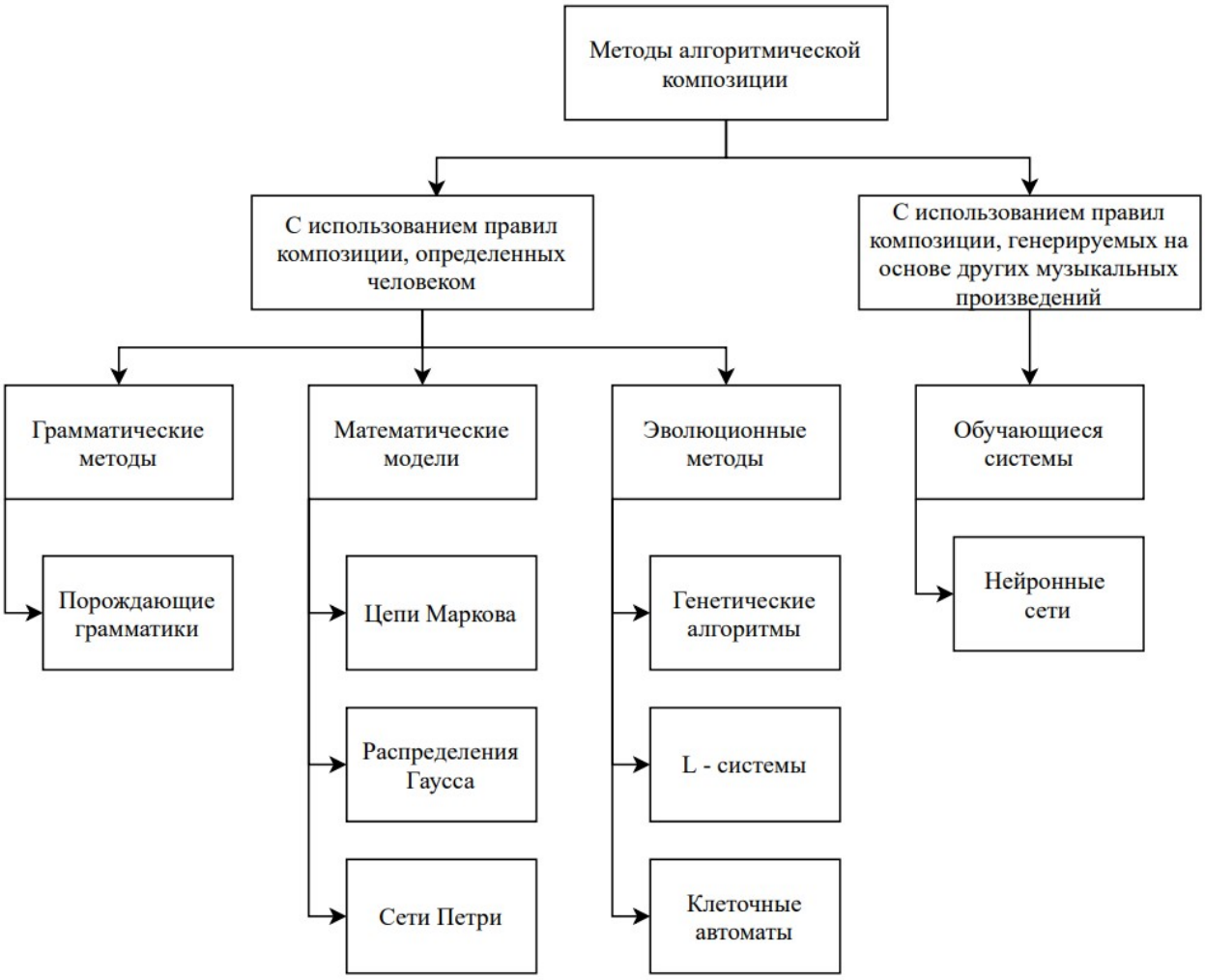

Рисунок 1.1 – Классификация методов алгоритмической композиции

Программам, основанным на одной алгоритмической модели, редко удается добиться эстетически удовлетворительных результатов. По этой причине алгоритмы разных типов часто используются вместе, чтобы объединить сильные и уменьшить слабые стороны этих алгоритмов. Стоит отметить, что серьезной проблемой гибридных систем является их растущая сложность и потребность в ресурсах для объединения и тестирования этих алгоритмов.

Для более подробного рассмотрения и дальнейшего сравнения выделим следующие методы, которые являются наиболее яркими представителями своих групп:

— порождающие грамматики;

— системы Линденмайера (Л-системы);

— цепи Маркова;

— искусственные Нейронные сети (ИНС).

Стоит отметить, что генетические алгоритмы, сети Петри и клеточные автоматы также являются достаточно значимыми методами алгоритмической композиции, но в данной работе они не будут рассматриваться в силу отсутствия экспертности оценки (с помощью них можно построить модель, но нельзя ее оценить, нужны критерии оценки, предиктивная модель).

1.2.1 Порождающие грамматики

Порождающая грамматика – формализм генеративной лингвистики, связанный с изучением синтаксиса [9]. В рамках подхода порождающей грамматики формулируется система правил, при помощи которых можно определить, какая комбинация слов оформляет грамматически правильное предложение.

Применение порождающих грамматик для задачи генерации музыкальных композиций является естественной процедурой, поскольку именно данный подход связан с формализацией, в том или ином виде, правил построения музыкальных композиций.

Порождающая грамматика задается кортежем, представленным ниже, на формуле (1.1):

$$G = \{N,\ T,\ P,\ S\}, \tag{1.1}$$

где

— N – конечный алфавит нетерминальных символов;

— T – конечный алфавит терминальных символов (совпадает с алфавитом языка, задаваемого грамматикой);

— P – конечное множество правил порождения;

— S – начальный нетерминал грамматики G.

Язык порождающей грамматики G – это множество цепочек, составленных из терминальных символов и порожденных из начального символа грамматики. Математическая формула (1.2) выглядит следующим образом:

$$L = w|S \to *w. \tag{1.2}$$

В рамках данного метода предполагается построение контекстно-зависимой грамматики по какому-либо обучающему набору. После построения грамматики, необходимо задать какую-то начальную мелодию и затем выводить композицию на основе построенных ранее правил. Для определения нот вполне естественно использовать их обозначение латиницей (C – нота «До», D – «Ре» и т. д.). Также в качестве алфавита можно использовать аккорды или применительно к MIDI-файлу – совокупности событий (к которым относится включение/выключение ноты, смена канала и т. д.).

Рассмотрим пример составления грамматических правил и генерации новой строки по созданным из строки (1.3) правилам:

$$ABCDEFGIKFHLEFJ. \tag{1.3}$$

Начнем строить грамматику для строки (1.3), начав, с символа F (данное действие нужно проделать для каждого символа). Необходимо написать правило, которое бы указывало, какую букву следует поставить, если вдруг встретился символ F. Лишь по одной букве невозможно определить что должно идти следом, поэтому к букве F добавляется контекст (символы, окружающие F). Проведя некоторые преобразования, получаются следующие конечные правила для буквы F, в зависимости от ее контекста выбирается какое-то одно правило из трех (1.4):

$$KF \to H, DEF \to G, LEF \to J. \tag{1.4}$$

Процесс генерации новой строки выглядит следующим образом: дана начальная последовательность, например, $ADEF$. Начинаем брать буквы с конца:

— F – нет правила с такой левой частью, расширяем контекст;

— EF – нет правила с такой левой частью, расширяем контекст;

— DEF – есть такое правило, ставим G, получаем $ADEFG$.

Затем данные действия проделываются сначала: берем букву G и т. д. столько раз, сколько необходимо.

Грамматику удобно представлять в виде дерева. Ниже, на рисунке 1.2, представлено дерево для буквы F:

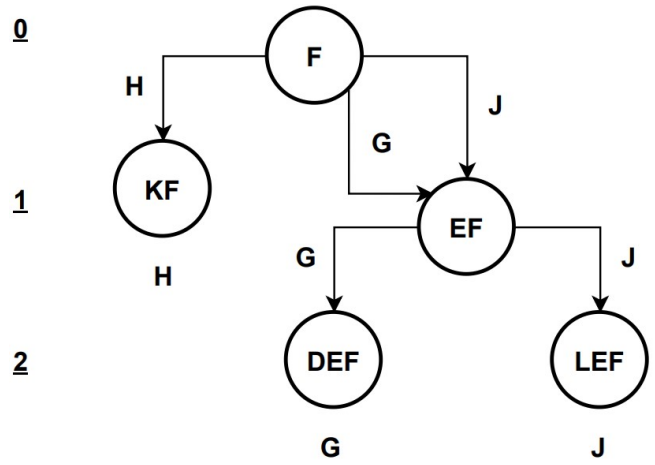

Рисунок 1.2 – Грамматика, представленная в виде дерева

В узлах находятся левые части правил, рядом с узлами написаны правые части правил – возможные продукции. Числа слева от дерева указывают, на каком контекстном уровне находятся узлы.

Преимуществом построения новой мелодии на основе уже существующей состоит в том, что не нужно самим придумывать правила построения, так как они формируются из мелодии, однако сам процесс определения чётких формальных правил построения композиции является крайне трудоёмким и требует большого количества входных данных. Кроме того, построение грамматических правил не гарантирует соблюдение всех музыкальных норм построения композиций, поскольку зачастую они гораздо сложнее чем прямая зависимость от последовательности предшествующих нот.

1.2.2 Л-системы

Одним из эволюционных методов решения задач алгоритмической композиции являются системы Линденмайера (далее Л-системы), лежащие на стыке таких сфер математики, как эволюционные методы и формальная грамматика [10].

В основе работы Л-систем лежит набор правил замещения, рекурсивно применяющийся на начальную строку символов и интерпретирующий конечную строку, как структурные элементы организма. Правила замещения определяют, как каждый конкретный символ в текущем поколении должен быть перемещен.

Базовое описание контекстно-независимой системы заключается в формуле (1.5):

$$G = |A,\ P,\ \alpha|, \tag{1.5}$$

где

— A – алфавит системы (набор всех символов включая пустой символ, ε);

— P – это конечной набор правил замещения (определенный в виде входящий символ – выходящий символ);

— α – аксимомы, символ или строка из алфавита, используемая в качестве начального состояния.

Применение Л-систем для генерации алгоритмических композиций подразумевает использование вместо символов формулы определенных музыкальных параметров. Например, A – алфавит из семи нот, входящих в гамму «До мажор», P – классические законы гармонии, либо правила, которые можно определить исходя из матрицы переходных вероятностей, α – начальная нота или аккорд.

В качестве примера можно привести простую систему, имитирующую рост водорослей Lémna на стоячей воде, также впервые смоделированную

Аристидом Линденмайером:

— A (альфавит) – $\{A, B\}$;

— P (правила замещения) – $(A \to AB), (B \to A)$;

— α (начальный символ) – A.

В итоге получаем последовательность поколений, представленную в таблице 1.2.

Таблица 1.2 – Последовательность поколений водорослей Lémna на стоячей воде

Поколение	Состояние
0	A
1	AB
2	ABA
3	ABAAB
4	ABAABABA
5	ABAABABAABAAB
6	ABAABABAABAABABAABABAABAABABAABAAB

Основным отличием Л-систем от формальных грамматик состоит в том, что правила применяются одновременно ко всей строке, к каждому символу. Также, нет понятий терминальных и нетерминальных символов, следовательно, «вывод» по этой грамматике может продолжаться бесконечно, что хорошо подходит для генерации музыкальной последовательности в режиме реального времени.

Одним из ярких примеров применения Л-систем для генерации музыки является программа LMUSE [2].

1.2.3 Цепи Маркова

Цепи Маркова являются распространенным и довольно простым способом моделирования случайных событий. Они используется в самых разных областях, начиная генерацией текста и заканчивая финансовым моделированием.

Основой данного метода является идея о том, что музыка вероятностна: определенные последовательности нот/аккордов более вероятны, чем другие, в зависимости от текущей последовательности аккордов. Из-за этой вероятностной природы метод цепей Маркова является «естественным» выбором для генерации музыкальной композиции.

Марковские цепи – это последовательности случайных величин, для которых вероятность появления того или иного значения на $(k+1)$-м шагу зависит лишь от того, какое значение эта величина приняла на k–м шагу, и не зависит от значений величины на 1-м, 2-м, ..., $(k-1)$-м шагах. Характеризуется тем свойством, что, говоря нестрого, при фиксированном настоящем будущее независимо от прошлого.

Конечная дискретная цепь определяется:

— Множеством состояний S. Событием является переход из одного состояния в другое в результате случайного испытания.

$$S = \{s_1,\ s_2,\ ...,\ s_n\}. \tag{1.6}$$

— Вектором начальных вероятностей $P(0)$ (начальным распределением):

$$P(0) = \{P(0)(1),\ P(0)(2),\ ...\ ,\ P(0)(n)\}, \tag{1.7}$$

где $P(0)(i)$ – вероятности того, что в начальный момент времени $t = 0$ процесс находился в состоянии s_i.

— Матрицей переходных вероятностей P, характеризующей вероятность

перехода процесса с текущим состоянием s_i в следующее состояние s_j, при этом сумма вероятностей переходов из одного состояния равна 1.

$$P = \{p_{\text{ij}}\}, \sum_{j=1}^{n} p_{ij} = 1. \tag{1.8}$$

Данный метод требует первоначального анализа некоторых существующих композиций для составления матрицы переходных вероятностей. Возьмем в качестве входных данных набор нот из детской английской песенки «Mary had a little lamb» и составим для нее соответсвующую марковскую модель 1-го порядка [11].

Исходная последовательность нот (1.9) для $S = \{C, D, E, F, G\}$:

$$EDCDEEEDDDEGGEDCDEEEEDDEDC. \tag{1.9}$$

Ниже, на рисунке 1.3, представлена матрица переходных вероятностей для данной последовательности. Каждый элемент P_{ij} матрицы представляет вероятность перехода из текущего состояния i в состояние j. Пустые ячейки представляют вероятность перехода 0.

Следующее событие

Текущее событие	C	D	E	F	G
C		2/3	1/3		
D	3/10	3/10	4/10		
E		5/11	5/11		1/11
F					
G			1/2		1/2

Рисунок 1.3 – Представление марковской модели 1-го порядка в виде матрицы переходных вероятностей

Также марковская цепь может быть представлена графом, вершины

которого соответствуют состояниям цепи, а дуги – ненулевым вероятностям переходов. Граф, соответсвующий рассматриваемой марковской модели, представлен на рисунке 1.4.

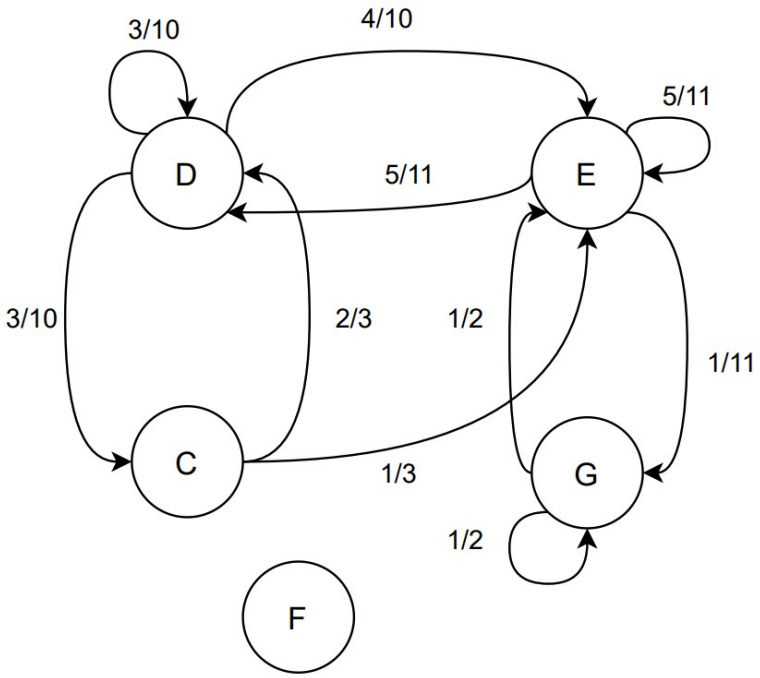

Рисунок 1.4 – Представление марковской модели 1-го порядка в виде графа переходов

Далее для генерации псоледовательности на основе составленной марковской модели необходимо случайным образом выбрать начальное состояние и решить для него задачу нахождения оптимального (наиболее вероятного) пути по матрице переходных вероятностей.

Стоит отметить, что чаще всего на практике используются марковские моделей более высоких порядков n, где вероятность наступления события зависит от n предыдущих, а также Скрытые Модели Маркова. Благодаря этому сгенерированная музыкальная композиция получается более сложной и многомерной, какой и являетя музыка на самом деле.

Марковские модели хорошо подходят для определённых музыкальных задач, например, для задачи имитации стиля. Тип и качество выходной композиции будет в значительной степени зависеть от свойств корпуса (набора), и может быть хорошо предсказано, в отличие от других процедур,

таких как нейронные сети.

1.2.4 Нейронные сети

Искусственные нейронные сети – вычислительная модель, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. Она состоит из связанного набора искусственных нейронов – очень простых вычислительных узлов, которые агрегируют несколько числовых входов в один выход. Нейроны сгруппированны в слои, ИНС состоит из входного слоя, одного или нескольких скрытых слоев и выходного слоя. Каждый нейрон связан с нейронами предыдущего слоя через определенные веса. Входной слой получает информацию, затем n скрытых слоев обрабатывают ее и выходной слой выводит результат. На рисунке 1.5, представлена схема однослойной ИНС.

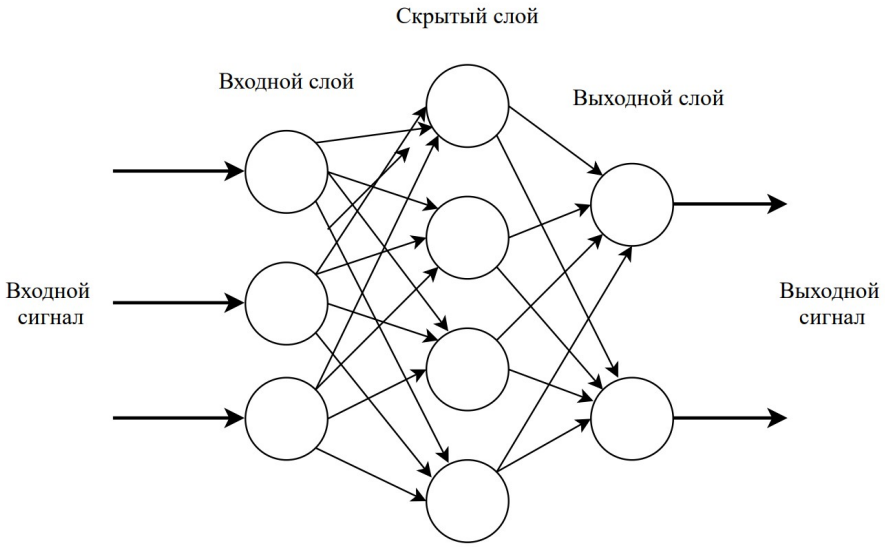

Рисунок 1.5 – Схема однослойной ИНС

Количество слоев и нейронов в ИНС не ограничено. Эта особенность позволяет нейронной сети моделировать очень сложные закономерности, с которыми бы не справились, например, линейные модели.

Для того, чтобы нейронная сети была способна выполнить поставленную задачу, ее необходимо обучить. Процесс обучения нейронной сети построен на

настройке весов между нейронами посредством анализа заранее заданного обучающего набора данных. Впоследствии эти веса используются для «предсказания» выходов по неразмеченным данным. Для обучения нейронных сетей существуют различные алгоритмы, однако самым распространённым является метод обратного распространения ошибки. Данный метод изменяет веса в зависимости от найденной ошибки, которая представляет собой разницу между ожидаемым и фактическим выходом нейронной сети. Изменение весов происходит в обратном направлении – от выходного слоя, где как раз можно определить наличие отклонения в ожидаемом и фактическом значении, затем продолжается к скрытым слоям и наконец до входного слоя [12].

На текущий момент работы, направленные на автоматизацию процесса генерации музыки, исследуют возможность генерации композиций с использованием нейронных сетей, поскольку именно данный метод позволяет выявить наиболее сложные взаимосвязи, характерные музыкальным произведениям, которые не видны человеку явно.

Для задачи генерации музыкальной композиции чаще всего используются Рекуррентные Нейронные Сети, в особенности LSTM (Long Short-term Memory), так как они позволяют запоминать предыдущую последовательность нот и предсказывать следующую последовательность, что оказывается полезным при генерации музыки, так как при добавлении в музыку следующего такта, требуется учитывать ранее сгенерированную последовательность нот.

Ниже, на рисунке 1.6, представлен фрагмент рекуррентной нейронной сети A: он принимает входное значение x_t и возвращает значение h_t. Наличие обратной связи позволяет передавать информацию от одного шага сети к другому.

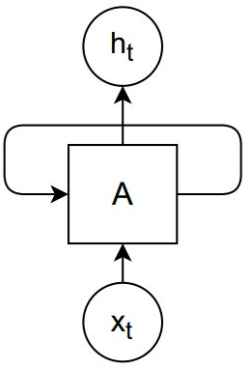

Рисунок 1.6 – Фрагмент рекуррентной нейронной сети

Рекуррентную сеть можно также рассматривать, как несколько копий одной и той же обычной нейронной сети, каждая из которых передает информацию последующей копии. На рисунке 1.7 представлена рекуррентная нейронная сеть в развертке.

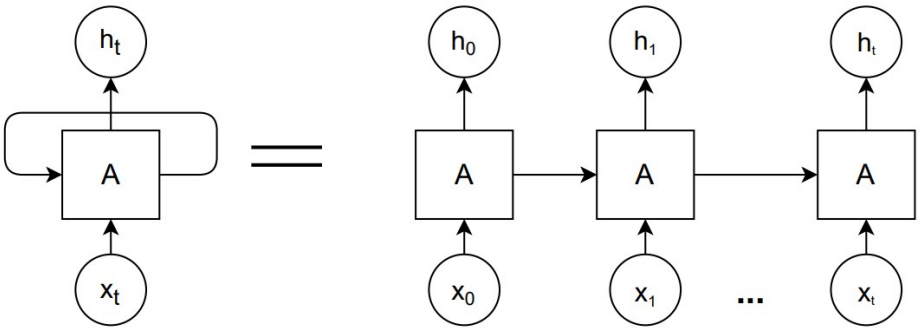

Рисунок 1.7 – Рекуррентная нейронная сеть в развертке

Стоит также отметить, что ИНС часто используются не только для генерации музыкальной композиции, но и для решения задач извлечения музыкальных паттернов из заданных композиций.

Основным преимуществом ИНС перед вышеперечисленными методами – это возможность генерировать новые комбинации из нотных последовательностей, не встречавшиеся в исходных данных.

Данный метод генерации музыкальных композиций используют программы Magenta [13], BachBot [3], DeepJazz [4] и многие другие.

1.2.5 Критерии сравнения

В результате классификации методов генерации музыкальной композиции были рассмотрены следующие методы:

— Порждающие грамматики;

— Л-системы;

— Цепи Маркова;

— Искусственные Нейронные сети.

Проведем сравнение вышеперечисленных методов с целью выбрать наиболее подходящий для генерации музыкального фрагмента в соответствии с эмоциональным состоянием человека.

Для сравнения методов выделим следующие критерии:

1. Достаточность малой обучающей выборки (отсутствие необходимости формирования большой обучающей выборки данных для обучения модели):
    — 1 – малой обучающей выборки недостаточно для обучения модели.
    — 5 – малой обучающей выборки достаточно для обучения модели.
2. Необходимость преобразования обучающего набора данных (готовых музыкальных композиций) и вывода правил для обучения модели:
    — 1 – построение правил и обучение модели производится вручную.
    — 2 – поддерживается обучение на готовых композициях с совместным использованием других методов.
    — 3 – поддерживается обучение модели на готовых композициях с существенным преобразованием данных.
    — 4 – поддерживается обучение модели на готовых композициях с незначительным преобразованием данных.

— 5 – преобразование данных и вывод правил не требуются, поддерживается обучение модели на готовых композициях.

3. Наличие фиксированной структуры музыкальной композиции:

— 1 – сгенерированный музыкальный фрагмент полностью повторяет стуктуру обучающей выборки.

— 3 – в структуре сгенерированного музыкального фрагмента присутствуют паттерны из обучающей выборки.

— 5 – структура сгенерированного музыкального фрагмента полностью отличается от структуры обучающей выборки.

4. Возможность изменения условий:

— 1 – метод не позволяет измененять условия.

— 3 – метод позволяет изменять только начальные условия.

— 5 – метод позволяет изменять условия в процессе генерации.

Ниже, на рисунке 1.8, представлено сравнение наиболее используемых методов алгоритмической композиции по определенным выше критериям:

Метод / Критерий	Л-системы	Искуственные Нейронные сети	Порождающие грамматики	Цепи Маркова
Достаточность малой обучающей выборки	5	1	5	5
Необходимость преобразования данных и вывода правил для обучения модели	3	4	1	3
Наличие фиксированной структуры музыкальной композиции	3	5	1	3
Возможность изменения условий	3	5	3	5

Рисунок 1.8 – Сравнение методов генерации музыкальной композиции

1.2.6 Выбор метода

Каждый из рассмотренных методов имеет свои недостатки, но главным отрицательным свойством практически всех вычислительных систем является то, что музыка, которую они продуцируют, лишена смысловой нагрузки – компьютеры не имеют чувств и настроений, не вкладывают в музыку эмоции, которые свойственны человеку.

Искуственные нейронные сети предоставляют очень хороший результат при использовании их для генерации музыки, но они очень сложны в создании и обучении (если же ИНС самообучаема, сложность её создания будет ещё выше).

Порождающие грамматики генерируют достаточно однотипные мелодии, к тому же они требуют определения чётких формальных правил построения композиции, что является крайне трудоёмким процессом.

Для Л-систем также как и для порождающих грамматик невозможно описать все правила перехода, необходимые для построения мелодии. К тому же поведение Л-систем трудно предвидеть, оно реагирует с высокой степенью чувствительности на происходящие изменения начальных условий, что дает широкий диапазон результирующих мелодий, но одновременно делает невозможным какое-либо прдесказание результата.

Марковские модели треубют достаточно большого объема данных для создания матрицы вероятностей переходов (чем больше входных данных, тем более разнообразной получится выходная мелодия), но в отличии от вышеперчисленных методов, их поведение может быть хорошо предсказано, что является важным критерием для данной работы.

В результате проведенного сравнительного анализа методов алгоритмической композиции в качестве метода генерации музыкального фрагмента в соответствии с эмоциональным состоянием человека будут использоваться марковские модели.

1.3 Связь звука и цвета

Звук представляет собой гармонические колебания, частоты которых относятся как целые числа и вызывают у человека приятные ощущения (консонанс). Близкие, но отличающиеся по частоте колебания вызывают неприятные ощущения (диссонанс). Звуковые колебания со сплошными спектрами частот воспринимаются человеком как шум.

Имеем следующие характеристики:

— частота, измеряется в герцах, она отображает, сколько раз в секунду происходит колебание;

— длина волны – величина обратная частоте, определяет промежуток между колебаниями. Между частотой и длинной волны существует связь:

$$\nu = \frac{V}{\lambda}, \tag{1.10}$$

где ν – частота, V – скорость волны, λ – длина волны.

Каждая нота имеет свою частоту, а каждый монохроматический (чистый) цвет определяется своей длиной волны, и, соответственно, тоже имеет частоту.

Два звука, частоты колебания которых относятся как 2:1, образуют чистую октаву. Нота находится на определённой октаве. Чтобы поднять ноту на одну октаву вверх, её частоту надо умножить на 2. Например, если нота «Ля» первой октавы имеет частоту 220 КГц, то у «Ля» второй октавы частота будет $220 \cdot 2 = 440$ КГц.

Если идти всё выше и выше по нотам, то можно заметить, что на 41 октаве частота будет попадать в спектр видимого излучения, который находится в диапозоне от 380 до 740 нанометров. На этом этапе можно начать сопоставлять ноту определённому цвету.

Весь видимый человеческим глазом спектр расположен в диапазоне одной октавы от «Фа диез» до «Фа» (от 400 до 790 ТГц). Начинается эта октава с красного и заканчивается фиолетово-красным цветом (голубой и синий цвета идентичны для эмоционального восприятия, разница только в интенсивности окраски). Октаву видимого света можно назвать цветовой. Следовательно тот факт, что человек выделяет в радуге 7 основных цветов, а в стандартной гамме 7 нот – это не просто совпадение, а взаимосвязь.

Ниже, в таблице 1.3, представлена схема соотношения цветов и нот.

Таблица 1.3 – Таблица соотношения между звуком и цветом

Нота	**Частота (Гц)**	**Цвет**	**Частота (Тгц)**	**Длина волны (нм)**
Фа диез (F#)	185,0	Красный	406	735
Соль (G)	196,0	Красно-оранжевый	431	693
Соль диез (G#)	207,0	Оранжевый	456	654
Ля (A)	220,0	Оранжево-желтый	483	619
Си бемоль (B#)	233,08	Желтый	512	585
Си (B)	246,98	Желто-зеленый	543	552
До (C)	261,63	Зеленый	575	521
До диез (C#)	277,18	Зелено-синий	609	492
Ре (D)	293,66	Синий	645	464
Ре диез (D#)	311,13	Сине-фиолетовый	684	438
Ми (E)	329,63	Фиолетовый	724	413
Фа (F)	349,23	Фиолетово-красный	767	390

1.4 Тест Люшера

Для определения эмоционального состояния человека предлагается использование специализированного теста Люшера [14], который является одним из психологических тестов для определения текущего и желаемого состояния человека путем анализа взаиморасположения предпочтительных цветов. При этом, во внимание берется порядок выбора и расположение так

называемых основных цветов (красный, синий, зеленый и желтый). Если все основные цвета расположены на главных позициях (до пятой), то можно говорить, что в той или иной мере выполняются потребности, описанные цветами. Если же какой-либо из основных цветов расположен дальше, то можно говорить о наличии конфликта.

Существует две разновидности теста Люшера:

1. Полный вариант цветового теста Люшера («Клинический цветовой тест»). Он состоит из восьми цветовых таблиц.
2. Сокращенный тест Люшера. Он использует ряд из восьми цветов.

В данной работе для определения эмоционального состояния человека будет использоваться сокращенный восьмицветовой тест Люшера.

Основные цвета:

— синий – символизирует спокойствие, удовлетворенность, безопасность, расслабление;

— зеленый – чувство уверенности, настойчивость, иногда упрямство;

— красный – символизирует силу волевого усилия, агрессивность, возбуждение, энергичность;

— желтый – активность, стремление к общению, желание действовать, веселость.

При отсутствии конфликта в оптимальном состоянии основные цвета должны входить в первые пять позиций. Дополнительные цвета (фиолетовый, коричневый, черный и серый) символизируют негативные тенденции: тревожность, стресс, переживание страха, огорчение.

Сама процедура тестирования состоит в упорядочивании испытуемым цветов по степени их субъективной приятности. В результате теста Люшера получается список цветов, расположенных в порядке предпочтения пользователя

в данный момент времени. Для определения эмоционального окраса генерируемого музыкального фрагмента необходимо проанализировать взаимное расположение основных цветов:

1. Если все основные цвета расположены на первых пяти позициях, то считается, что потребности описываемых цветов удовлетворены. Тогда:

   — генерирумый музыкальный фрагмент будет иметь мажорное наклонение;

   — по доминирующему основному цвету с использованием таблицы соотношения цветов и нот определяется тоника музыкального фрагмента.

2. Если основные цвета занимают позицию далее пятой, значит, характеризуемые ими свойства, потребности не удовлетворены, следовательно, имеют место тревожность, негативное состояние. В этом случае:

   — генерируемый фрагмент будет иметь минорное наклонение;

   — по отвергаемому основному цвету с использованием таблицы соотношения цветов и нот определяется тоника музыкального фрагмента.

Таким образом, имея тонику и лад, можно определить тональность генерируемого музыкального фрамента.

1.5 Структура MIDI файла

MIDI (Musical Instrument Digital Interface) – стандарт цифровой звукозаписи, формат обмена данными (интерфейс) между электронными музыкальными инструментами. Данный интерфейс позволяет единообразно кодировать в цифровой форме такие данные как нажатие клавиш, настройку громкости и других акустических параметров, выбор тембра, темпа, тональности и т. д., с точной привязкой во времени.

MIDI файл состоит из:

1. Заголовка, его структура показана на рис.1.9. В поле длины всегда содержится число 6 – по числу байт данных заголовка, следующих за этим полем. Данные заголовка представляют собой три 16-битных слова. Заголовок хранит информацию о:

   — Формате MIDI-файла (format). Сущетсвует три формата MIDI-файлов, они задаются цифрами 0, 1, 2 и задают количество треков, которое может содержаться в Midi-файле. Трек – это отдельная часть файла, которая содержит информацию о музыкальной композиции. Каждый трек содержит информацию о нотах, темпе, громкости и других параметрах, которые определяют звучание музыки. В файлах формата 0 все MIDI события и данные расположены в одном треке, файлы формата 1 могут включать любое количество треков, которые проигрываются одновременно. Файлы формата 2 практически не используются, для них характерно асинхронное проигрывание треков.

   — Число блоков MIDI-файла (ntrks). Данный параметр актуален лишь для форматов 1 и 2, так как в формате 0 блок всегда 1.

   — Формат времени MIDI-файла (division). Существуют музыкальный (задается количеством тиков на четвертную ноту) и абсолютный (показывает количество тиков в SMPTE блоке) форматы времени. Структура данного поля показана на рисунке 1.10.

Блок заголовка		
Заголовок		**Данные**
ID	**Длина блока**	
'MThd'	00 00 00 06	format ntrks division

Рисунок 1.9 – Структура блока заголовка

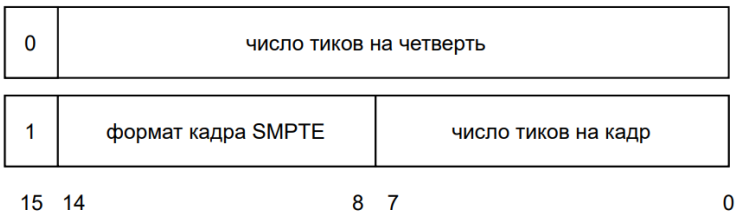

Рисунок 1.10 – Формат времени MIDI файла

2. Блоков трека, в который в хронологическом порядке записаны MIDI-события, каждое из которых является командой для музыкального инструмента. В данном блоке должно присутствовать хотя бы одно событие. Структура блока трека представлена на рисунке 1.11.

Блок трека		
Заголовок		**Данные**
ID	**Длина блока**	
'MTrk'	nn nn nn nn	событие 1, событие 2, ..., событие n

Рисунок 1.11 – Структура блока трека

MIDI-событие (рис.1.12) состоит из дельта-времени (измеряется в тиках и показывает время, прошедшее с момента наступления предыдущего события), и самого сообщения MIDI.

Дельта-время	**Сообщение MIDI**	
ticks	статус байт	байты данных

Рисунок 1.12 – Структура MIDI-события

MIDI-события делятся на две большие группы:

— Системные или мета-события, они передают команды, которые воздействуют на общие параметры и режимы работы всех устройств-получателей, и позволяют манипулировать мета данными, относящимися к MIDI-файлам. Данные события начинаются со статус-

байта 0xFF, затем идет тип мета-события, а после – длина данных и сами данные (рис. 1.13).

Статус-байт	Тип мета-события	Длина	Данные
FF	tt	...	...

Рисунок 1.13 – Структура мета-события

— канальные (повторяют структуру рис.1.12), они включают в себя номер MIDI-канала и передают сообщения на каждый MIDI-канал индивидуально. Всего существует 16 каналов, каждому из них назначается конкретный инструмент и тембр (канал 10 по традиции используется для ударных инструментов). Данные канальных событий сопровождаются следующей информацией: channel (номер канала), pitch (высота, название ноты с октавой) и velocity (скорость взятия ноты).

Канальные события, которые представляют наибольший интерес в данной работе:

— Note Off (0x8) – событие выключения ноты (клавиша отпущена);

— Note On (0x9) – событие включения ноты (нажатие на клавишу).

Также в данной работе представляют интерес следующие мета-события:

— Set Tempo (0x51) – задает текущий темп в микросекундах на четверть.

— Time Signature (0x58) – задает музыкальный размер композиции, который определяется числителем (numerator, поле nn) и знаменателем (denominator, поле dd), который является степенью числа 2. Поле cc задает число сообщений MIDI Clock, приходящихся на один удар метронома. Поле bb – это число тридцать вторых нот, приходящихся на 24 сообщения MIDI Clock (то есть на четверть). Например, событие Time Signature при размере 6/8, клике метронома на каждые три восьмых и 24-х MIDI Clock в четверти будет выглядеть как 0xFF 0x58 0x04 0x06 0x03 0x24 0x08.

— Key Signature (0x59) – задает текущую тональность. Поле sf содержит количество ключевых знаков в тональности (от -7 до $+7$, отрицательные значения – бемоли, положительные – диезы, нулевое значение – тональность «До мажор» или «Ля минор», т.е. без знака). Поле mi уточняет наклонение – 0 или 1.

Структура перечисленных выше мета-событий представлена на рис.1.14.

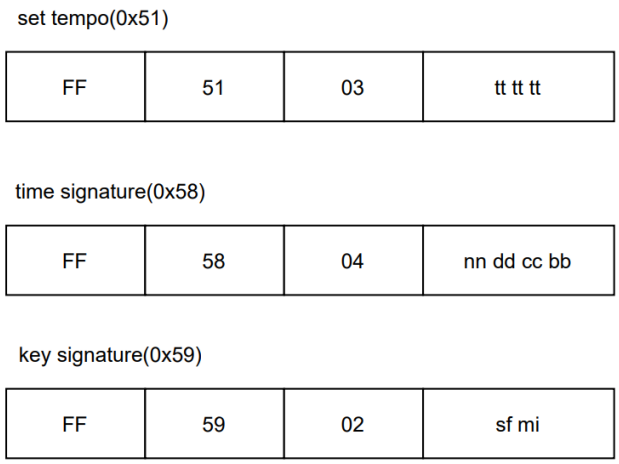

Рисунок 1.14

1.6 Формализация задачи

Суть задачи – сгенерировать музыкальный фрагмент, соответсвующий эмоциональному состоянию человека. Ниже, на рисунке 1.15, представлена формализация задачи в виде IDEF0-диаграммы нулевого уровня.

Рисунок 1.15 – IDEF0-диаграмма задачи генерации музыкального фрагмента

Входные данные включают в себя:

1. результат теста Люшера (список из 8 цветов, расположенных в порядке препочтения пользователя в данный момент времени);
2. длительность генерируемого музыкального фрагмента (задается количеством тактов, которые необходимо сгенерировать).

На выход метод выдает MIDI-файл со сгенерированным музыкальным фрагментом.

Поставленная задача решается в 3 этапа:

1. предварительная обработка входных данных;
2. генерация музыкального фрагмента;
3. преобразование реузльтата.

Рисунок 1.16 – Этапы решения задачи генерации музыкального фрагмента

Метод алгоритмической композиции с использованием марковских моделей требует предварительного обучения модели (создания обучающего набора данных). Данный этап проводится единожды и не требует повторения при каждой работе разрабатываемого метода.

Создание обучающего набора данных происходит в 2 этапа:

1. классификация MIDI-файлов;
2. анализ MIDI-файлов.

Ниже, на рисунках 1.17 - 1.18, представлены IDEF0-диаграммы данного этапа.

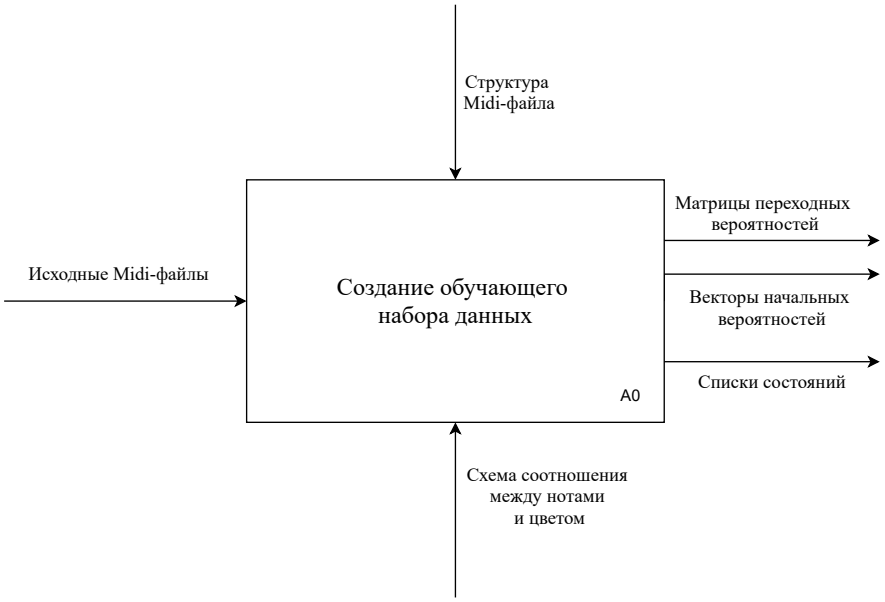

Рисунок 1.17 – IDEF0-диаграмма задачи создания обучающего набора данных

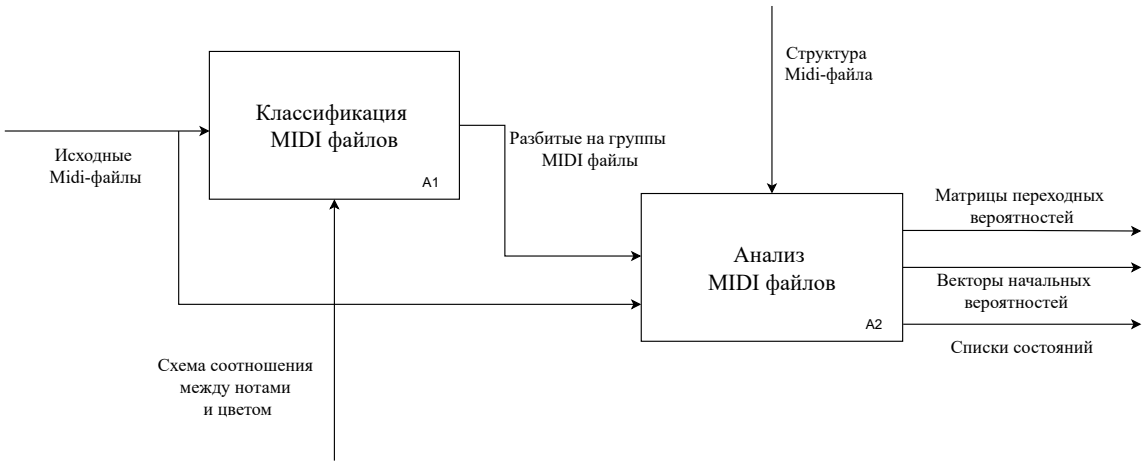

Рисунок 1.18 – Этапы создания обучающего набора данных

На решаемую задачу накладывается ряд ограничений:

— музыкальный фрагмент генерируется для одного инструмента – фортепиано;

— MIDI-файл с музыкальным фрагментом содержит только один трек (файл формата 0);

— сгенерированный музыкальный фрагмент не имеет жанровых особенностей;

— все генерируемые фрагменты имеют одинаковый музыкальный размер – 4/4;

— в сгенерированном фрагменте плотность звучания достигается за счет уплотнения музыкального текста.

1.7 Выбор базы данных

1.7.1 Реляционные БД

Реляционные БД – это системы, которые хранят данные в виде организованном по реляционной модели набора таблиц. Реляционная модель данных [15] является совокупностью данных и состоит из набора двумерных таблиц. При табличной организации отсутствует иерархия элементов. Таблицы состоят из строк – записей и столбцов – полей. На пересечении строк и столбцов находятся конкретные значения. Для каждого поля определяется множество его значений. В каждой таблице должно быть хотя бы одно ключевое поле, содержимое которого уникально для любой записи в этой таблице. Значения ключевого поля однозначно определяют каждую запись в таблице. Для построения логической связи между записями разных таблиц используются внешние ключи. За счет возможности просмотра строк и столбцов в любом порядке достигается гибкость выбора подмножества элементов. При использовании реляционной модели данных предполагается неделимость данных, хранящихся в полях записей таблиц. Также одним из основных положений реляционной модели данных является требование нормализации отношений.

Реляционные БД обычно используются для хранения и обработки структурированных данных, таких как данные банковских транзакций, учетные записи клиентов и другие типы данных, которые могут быть представлены в виде таблиц с определенными связями между ними.

Преимущества реляционных БД:

— простота и удобство использования;

— хорошо подходят для хранения структурированных данных с четкими связями между ними;

— обеспечивают высокую надежность и целостность данных благодаря использованию транзакций;

— поддерживают язык SQL для выполнения запросов к данным.

К недостаткам реляционных БД относят:

— неэффективны при работе с большими объемами неструктурированных данных, таких как тексты, изображения, видео;

— требуют строгой схемы данных, что затрудняет изменение структуры базы данных в процессе разработки;

— плохая масштабируемость, вызывающая стремительное падение производительности при росте объема данных и количества используемых в запросах соединений (JOIN) таблиц;

— неустойчивость к отказам оборудования.

1.7.2 Постреляционные БД

На рисунке 1.19 представлена классификация постреляционных БД по модели данных.

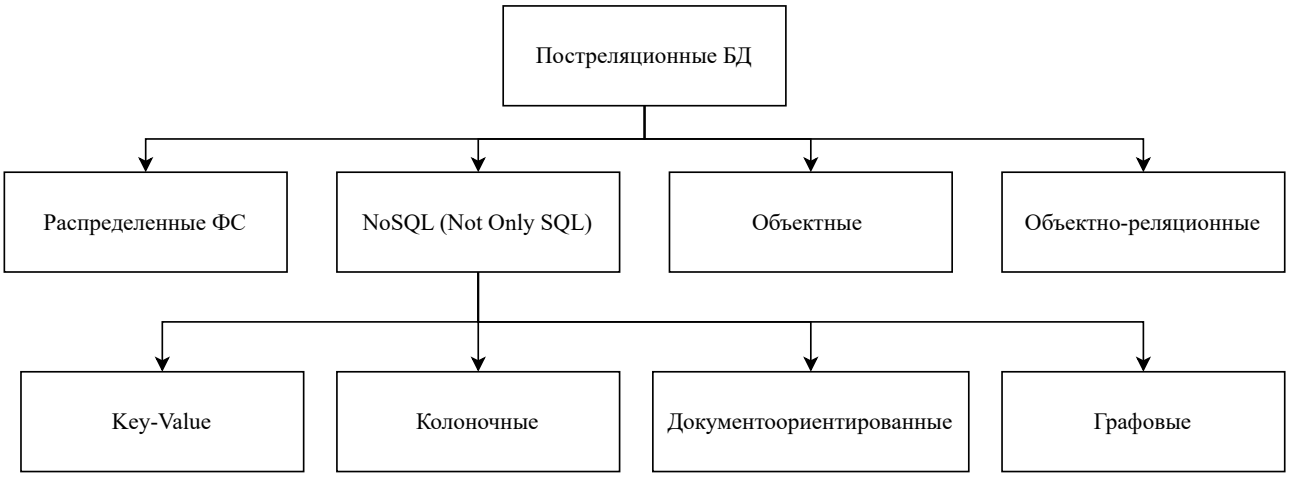

Рисунок 1.19 – Постреляционные БД

Существует несколько видов постреляционных БД [16], каждый из которых используется для хранения и обработки данных в различных форматах:

1. Распределенные файловые системы. Представляют собой системы, которые позволяют хранить и обрабатывать данные на нескольких компьютерах, объединенных в сеть. Распределенные ФС используются для обработки больших объемов данных, таких как аудио- и видеофайлы, данные о клиентах и транзакциях в банковских системах, а также для обработки данных в режиме реального времени.

2. NoSQL БД. Они используются для хранения и обработки неструктурированных данных, таких как данные социальных сетей, мультимедийные данные, данные IoT и другие типы данных, которые не могут быть представлены в виде таблиц с определенными связями между ними.

3. Объектные БД. Они хранят данные в виде объектов, которые могут быть связаны друг с другом, что позволяет создавать более сложные структуры данных. Объектные БД используются для хранения и обработки данных в приложениях, где важна высокая производительность и быстрый доступ к данным.

4. Объектно-реляционные БД (ОРБД). Данные системы, которые сочетают в себе возможности реляционных и объектных БД. Они позволяют хранить и обрабатывать данные в виде объектов, но также поддерживают SQL-запросы и традиционные реляционные таблицы. ОРБД используются для хранения и обработки данных в приложениях, где необходима гибкость объектной модели данных и возможность использования SQL-запросов.

NoSQL баз данных

По модели данных NoSQL-базы данных [17] классифицируются на следующие основные виды:

1. Ключ-значение (Key-Value). Хранят данные в виде пар ключ-значение, что делает их быстрыми и эффективными для операций чтения и записи. Они обеспечивают высокую производительность и масштабируемость, но могут быть менее гибкими в отношении запросов.

2. Документоориентированные (Document-Oriented). Используют модель данных, основанную на документах, что делает их подходящими для

хранения сложных иерархических данных. Данные БД обеспечивают хорошую масштабируемость и гибкость, но могут иметь проблемы с производительностью при выполнении сложных запросов.

3. Колоночные (Column-Family). Используют модель данных, основанную на столбцах, что делает их подходящими для хранения больших объемов структурированных данных. Они обеспечивают хорошую масштабируемость и производительность при выполнении сложных запросов, но могут быть менее гибкими в отношении схемы данных.

4. Графовые (Graph). Используют модель данных, основанную на графах, что делает их подходящими для хранения связанных данных, таких как социальные сети или графы знаний. Они обеспечивают хорошую производительность и гибкость при выполнении запросов, но могут быть менее эффективными в отношении масштабируемости.

NoSQL БД имеют ряд преимуществ:

— они позволяют эффективно работать с большими объемами неструктурированных данных;

— гибко подстраиваются под требования проекта и не требуют строгой схемы данных.

К недостаткам NoSQL БД относят меньшую надежность и целостность по сравнению с другими видами постреляционных БД, так как не все NoSQL БД поддерживают транзакции.

1.7.3 Выбор БД для хранения обучающего набора данных

Реляционные БД используют жесткую схему данных, что делает их менее гибкими в отношении изменения структуры данных, и могут иметь проблемы с масштабируемостью данных, что является недостатком для задачи хранения обучающего набора данных.

Так как обучающий набор данных имеет большой объем и не имеет фиксированной структуры, то для его хранения лучше всего подойдут NoSQL БД, а именно Key-Value хранилища, т.к. решаемая задача:

— требует быстрого времени отклика (т.е. выполнения операций чтения);

— не имеет четкой структуры данных.

Вывод

В данном разделе была описана предметная область алгоритмической композиции. Были проанализированы существующие методы генерации музыкального фрагмента, произведен сравнительный анализ, на основе которого был выбран наиболее подходящий для решения поставленной задачи метод. Также в данной разделе была приведена схема соотношения цвета и звука(нот), описан стандарт цифровой звукозаписи MIDI, выбран тип БД для хранения обучающего набора данных и формализована решаемая задача.

2 Конструкторский раздел

2.1 Inverse Transform Sampling

Мультиномиальное распределение – это обобщение биномиального распределения на случай $n > 1$ независимых испытаний случайного эксперимента с несколькими возможными исходами.

Пусть $X_1, \ldots, X_n$ – независимые одинаково распределённые случайные величины, такие что их распределение задаётся функцией вероятности 2.1:

$$P(X_i = j) = p_j, j = 1, \ldots, k. \tag{2.1}$$

Событие $X_i = j$ означает, что i-е испытание привело к исходу j.

Получаем, что мультиномиальное распределение появляется в последовательности зависимых случайных испытаний с конечным или счётным числом исходов. По сути – это цепь Маркова, где X_{i+1}-ая случайная величина зависима от предшествующей X_i-ой случайной величины следующим образом: X_{i+1}-ая случайная величина в t_{i+1}-ый момент времени принимает числовое значение, равное n_{i+1}, при условии, что в t_i-ый момент времени X_i-ая случайная величина приняла числовое значение, равное n_i.

Таким образом получаем, что совокупность переходов из любого заданного состояния цепи Маркова (т.е. соответствующая строка в матрице переходных вероятностей) формирует мультиномиальное распределение. Следовательно, можно проводить моделирование по цепи Маркова, моделируя по мультиномиальному распределению.

Один из способов моделирования на основе мультиномиального распределения – это выборка из обратного преобразования (inverse transform sampling). Данный метод позволяет выборку значений из любого распределения, для которого известен способ вычисления обратной функции для функции кумулятивного распределения.

Метод включает в себя 2 основных шага:

1. Сгенерировать u – равномерно распределенное случайное число от нуля до единицы включительно:

$$u \sim U(0, 1). \tag{2.2}$$

2. Вычисление результирующего значения с использованием кумулятивной функции F как

$$c = F^{-1}(u), \tag{2.3}$$

где

$$F(c) = u. \tag{2.4}$$

Грубо говоря суть метода заключается в том, чтобы представить функцию вероятностей как линию длиной 1, разделенную на интервалы, пропорциональные имеющимся вероятностям (для этого используется кумулятивная функция), а затем выбрать подходящий интервал на основе равномерно распределенного случайного числа.

Допустим, распределение дискретной случайной величины $X = \{1, 2, 3, 4\}$ задается функцией вероятности P:

$$P = \{p(x = 1) = 0.4, p(x = 2) = 0.1, p(x = 3) = 0.2, p(x = 4) = 0.3\}. \tag{2.5}$$

Функция кумулятивного распределения F (Cumulative distribution function, CDF):

$$F(C) = p(X <= C) = p(x = 1) + p(x = 2) + \cdots + p(x = C). \tag{2.6}$$

Данная функция обладает следующими свойствами:

— F(C) принимает значения в диапазоне [0, 1] и показывает количество случаев, в которых значение случайной величины X меньше C.

— F(C) монотонно возрастает.

— F(C) может быть использована как для дискретных, так и для непрерывных случайных величин.

Функцию вероятностей P представим как линию длиной 1, а значения этой функции как интервал на данной линии. На рисунке 2.1 голубая точка – это равномерно распределенное случайное число u, оно принимает значение 0.62, поэтому результатом выборки из P будет x=3, т.к. это следующее значение к точке справа (u попадает в интервал, соответсвующий x=3).

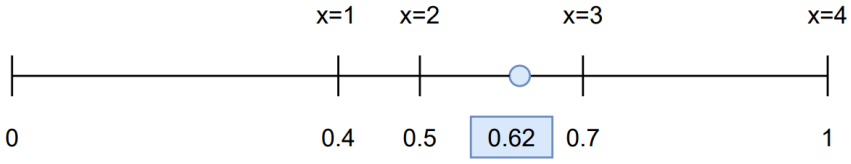

Рисунок 2.1 – Inverse Transform Sampling

2.2 Создание обучающего набора данных

2.2.1 Классификация данных

Марковская цепь, использующая матрицу переходных вероятностей, составленную на основе какого либо обучающего набора данных, может быть использована для генерации музыкального фрагмента, повторяющего стиль исходных данных. Следовательно, для решения задачи необходимо создание 8-ми обучающих наборов даных.

Имеющийся набор MIDI-файлов разделется на 8 групп, каждая из которых соответсвует паре [цвет, лад]. Ниже, на рисунке 2.2, представлены описанные ранее группы.

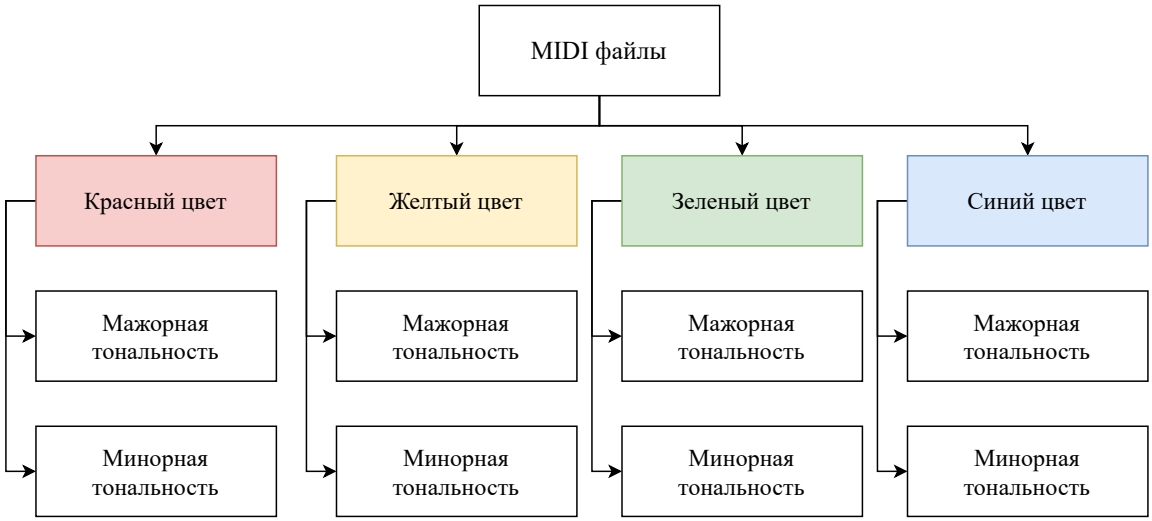

Рисунок 2.2 – Группы MIDI файлов

На основе таблицы 1.3 из предыдущего раздела определяется таблица соотношения между тоникой (первая ступень мажорного/минорного звукоряда) и цветом.

Нота	Цвет
Фа (F)	Красный
Фа Диез (F#)	Красный
Соль (G)	Красный
Соль Диез (G#)	Желтый
Ля (A)	Желтый
Си Бемоль (B#)	Желтый
Си (B)	Зеленый
До (C)	Зеленый
До Диез (C#)	Зеленый
Ре (D)	Синий
Ре Диез (D#)	Синий
Ми (E)	Синий

Рисунок 2.3 – Соотношение между тоникой и цветом

Определение принадлежности MIDI-файла к одной из групп происходит следующим образом:

1. Получить словарь тональностей, используемых в MIDI-файле (мета-событие key signature 0x59), в котором:
   - ключ – тональность, представленная буквой латинского алфавита (верхний регистр обозначает мажорную тональность, нижний – минорную);
   - значение – количество раз, которое данная тональность используется в MIDI-файле.
2. Определить доминирующую тональность в полученном словаре.
3. Определить тонику и лад доминирующей тональности.
4. С помощью таблицы, представленной на рис.2.3, определить цвет, которому соответсвует найденная ранее тоника.
5. На основе цвета и лада определить группу, к которой принадлежит MIDI-файл.

2.2.2 Анализ данных

Для генерации музыкального фрагмента необходимо, чтобы состояния цепи описывали определенные звуковые объекты. Звуковой объект может представлять либо отдельную ноту, либо аккорд (совокупность нот, сыгранных одновременно) и может быть однозначно определен:

- нотой, обозначается буквой латинского алфавита (от «A» до «G»), за которой следует знак ноты – «#» (диез), «b» (бемоль) или без знака;
- октавой, целое число в диапазоне [0, 8];
- длительностью, вещественное число в диапазоне [0, d], где d – знаменатель музыкального размера (denominator).

Например, звуковой объект, представляющий четвертную ноту «До Диез» в четвертой октаве, будет представлен как ($C\#4$ 0.25). Аккорд длительностью четверть такта из трех таких нот будет записан как (($C\#4, C\#4, C\#4$) 0.25). Также добавляется частный случай – пауза (обозначает, что звук не

производится). Она также характеризуется длительностью и обозначается латинской буквой R.

Множество состояний определяется путем анализа обучающего набора данных, который состоит из произведений для фортепиано, представленных в формате MIDI.

Звуковые объекты извлекаются последовательно из обучающего набора и записываются в:

1. Список states, который представляет собой множество состояний (уникальных звуковых объектов в обучающем наборе), список не содержит повторений.

2. Упорядоченный словарь states_appearances_dict, в котором ключ – это звуковой объект, а значение – количество появлений соответсвующего звукового объекта в обучающем наборе данных.

3. Упорядоченный словарь states_transition_dict, в котором хранится информация о том, сколько раз в обучающем наборе данных звуковой объект s_i переходит в звуковой объект s_j.

   Пример: $\{(F4\ 0.25) : \{(C\#3\ 0.5) : 2, (R\ 1) : 1\}\}$ – звуковой объект ($C\#3$ 0.5) дважды появляется после звукового объекта ($F4$ 0.25), а пауза (R 1) – появляется единожды после ($F4$ 0.25).

Таким образом гарантируется, что список состояний и словари содержат одни и те же звуковые объекты в одном и том же порядке, что позволит корректно сформировать матрицу переходных вероятностей.

Последнему звуковому объекту в каждом MIDI-файле обучающего набора данных присваивается переход в паузу длиной четверть такта, а самой паузе – переход в первый звуковой объект анализируемого файла. Данное действие гарантирует, что цепь Маркова не будет содержать поглощающих состояний, т.е. состояний, из которых нельзя перейти в другие.

2.2.3 Создание матрицы переходных вероятностей

Для составления матрицы переходных вероятностей необходимо определить вероятность появления каждой возможной пары звуковых объектов ($s1$, $s2$), т.е. вероятность перехода от $s1$ к $s2$ в цепи.

Пусть длина списка состояний (т.е. количество уникальный зуковых объектов в обучающем наборе) равняется n, тогда матрица переходов будет иметь размер $n \times n$. Матрица переходов составляется следующим образом:

1. Инициализация матрицы M размером $n \times n$ нулями.

2. Элементу $M_{i,j}$ матрицы присваивается количество раз, которое i-ый звуковой объект переходит в j-ый звуковой объект. На данном этапе матрица симметрична.

3. После того, как все элементы матрицы M были проинициализированы, каждый элемент строки i делится на сумму элементов строки i для получения вероятности соответствующего события.

4. Затем к каждой строке i матрицы M применяется кумулятивная функция, в результате чего первый элемент в строке $M_{i,0}$ сохранит свое значение с предыдущего шага и все последующие значения элементов строки будут последовательно возрастать. Значение последнего элемента в строке будет равняться единице $M_{i,n-1} = 1$.

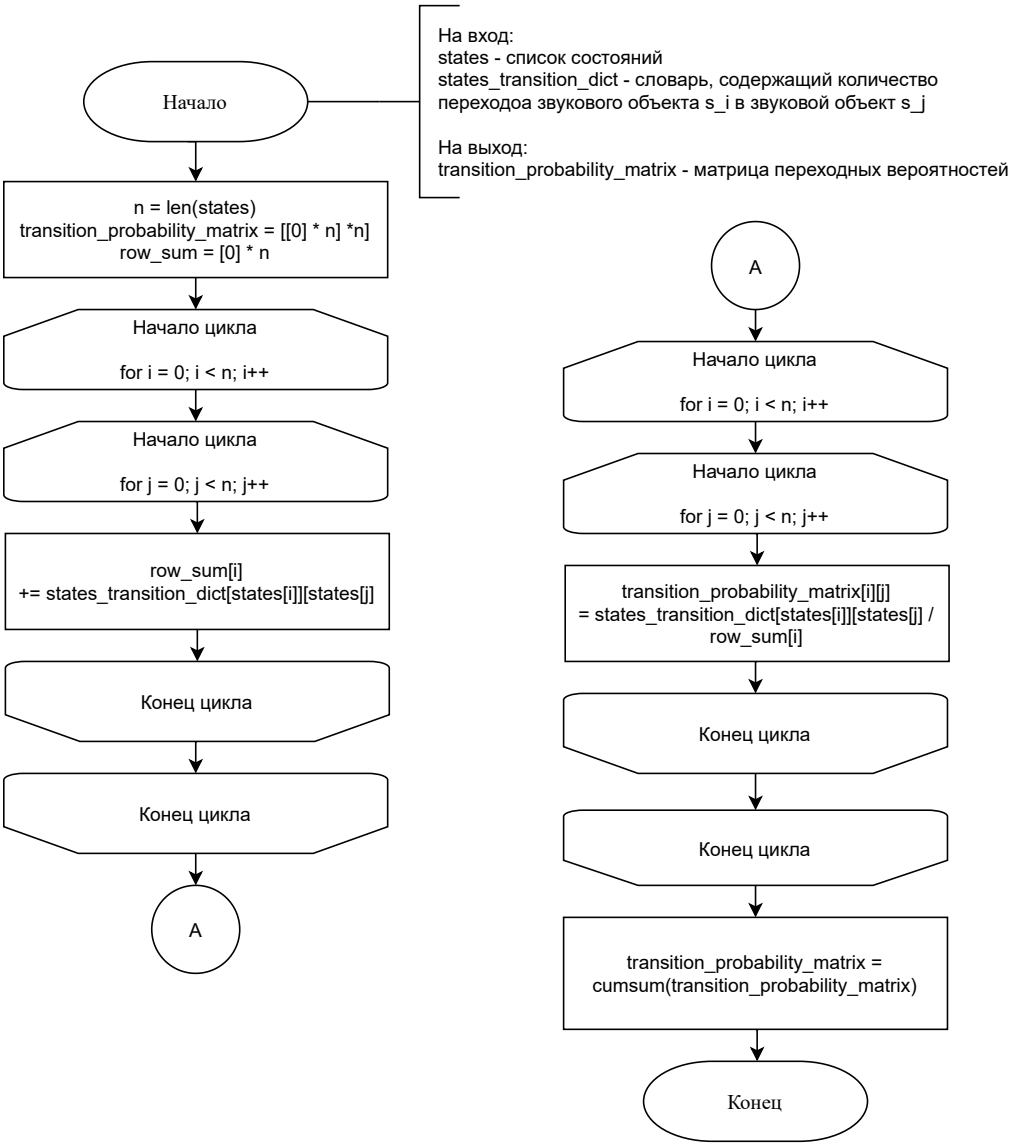

Рисунок 2.4 – Создание матрицы переходных вероятностей

Марковская цепь, использующая составленную матрицу может быть использована для генерации музыкального фрагмента, повторяющего стиль исходных данных. Поэтому для решения задачи создается 8 обучающих наборов, которые были описаны ранее.

2.2.4 Создание вектора начальных вероятностей

Вектор начальных вероятностей составляется аналогично:

1. Инициализация вектора V длиной n нулями, где n – количество различных состояний.

2. V_i элемент вектора инициализируется количеством появлений i-го состояния (звукового объекта) в обучающем наборе.

3. После инициализации всех n элементов вектора V, значение каждого элемента делится на сумму всех элементов вектора (т.е. на общее число звуковых объектов в обучающем наборе). Таким образом, мы получаем вероятность появления i-го состояния.

4. К полученному вектору применяется кумулятивная функция распределения, в результате чего первый элемент вектора V_0 сохраняет свое значение с предыдущего шага, а значения всех последующих элементов будут последовательно возрастать. Значение последнего элемента будет равно единице $V_{n-1} = 1$.

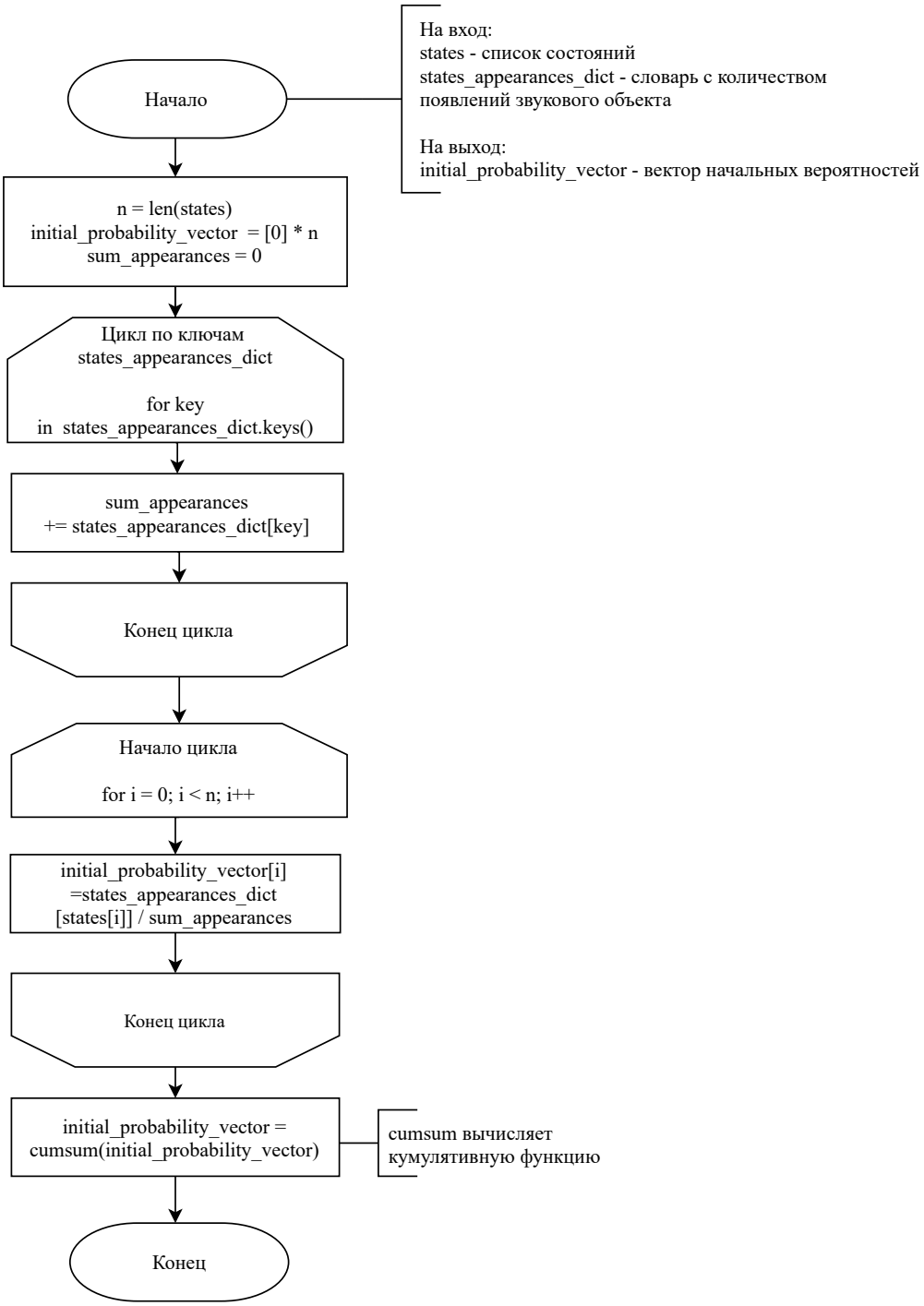

Рисунок 2.5 – Создание вектора начальных вероятностей

2.2.5 Хранение обучающего набора данных

Так как этап создания обучающего набора данных проводится единожды и не повторяется при каждой работе разрабатываемого метода, то необходимо сохранить результаты этапа (т.е. список состояний цепи Маркова, вектор начальных вероятностей и матрицу переходных вероятностей) для

дальнейшего использования.

Для хранения обучающего набора данных было решено использован Key-Value хранилища.Они используют в качестве БД ассоциативный массив, в котором отдельный ключ связан с одним объектом в коллекции, для которого ключ является уникальным идентификатором.

Ниже, на рисунке 2.6, представлены схемы алгоритмов записи и чтения объекта в key-value хранилищах.

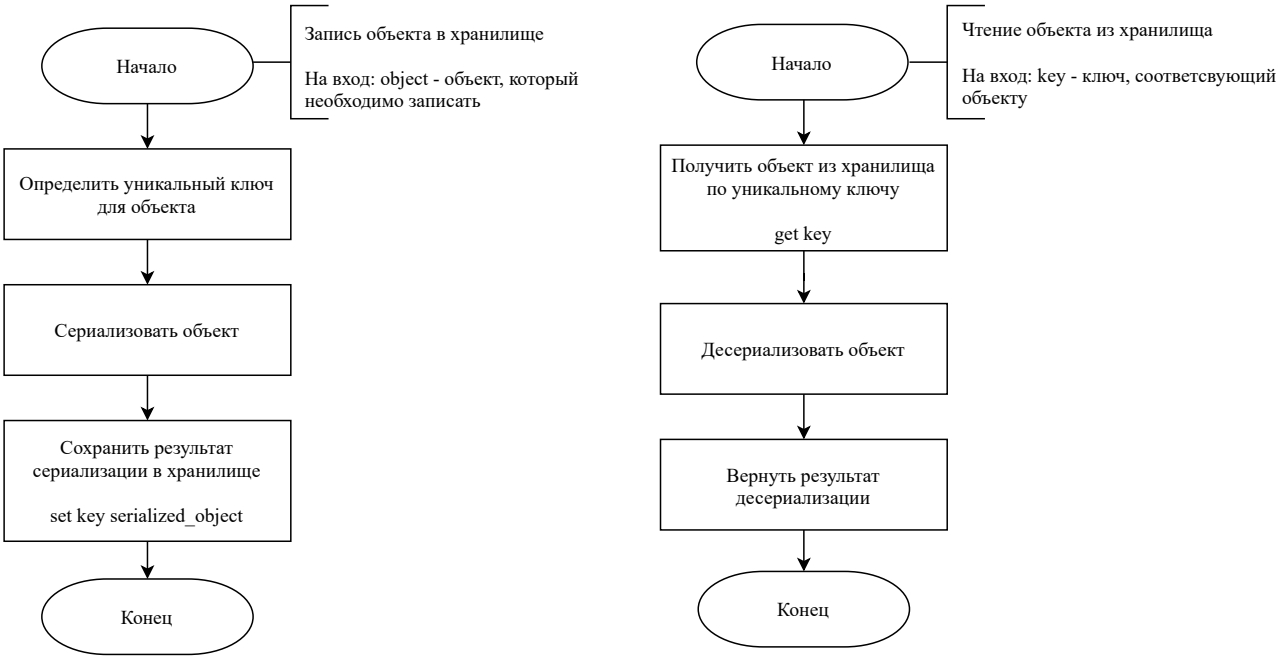

Рисунок 2.6 – Запись и чтение объекта

2.3 Обработка входных данных

В качестве входных данных этап получает результат теста Люшера, который представлен списком из 8-ми цветов, расположенных в порядке препочтения пользователя в данный момент времени, и длительность генерируемого фрагмента, которая задана количеством тактов. На выход данный этап выдает характеристики фрагмента в виде словаря meta, в котором:

— ключ meta['color'] соответсвует доминирующему/отвергаемому цвету в тесте Люшера;

— ключ meta['lad'] сответсвует ладу фрагмента;

— ключ meta['bpm'] определяет темп фрагмента в bpm (beats per minute).

По параметрам color и lad определяется обучающий набор данных, на основе которого будет произведена генерация музыкального фрагмента.

Определение характеристик музыкального фрагмента происходит следующим образом:

1. Проверка спика-результата теста Люшера $luscher\_result$ на вхождения основных цветов в первые 5 позиций.

2. Если условие выше выполняется, то доминирующий цвет определеяется как первый встретившийся основной цвет в списке $luscher\_result$. Ключу meta['color'] присваивается найденное значение, а ключу meta['lad'] – мажорный лад.

3. В случае нквыполнения условия определется отвергаемый цвет как первый встретившийся основной цвет в обращенном списке $luscher\_result$. Ключу meta['color'] присваивается найденное значение, а ключу meta['lad'] – минорный лад.

4. ключу meta['bpm'] присваивается случайно найденный темп в зависимости от выбранного обучающего набора данных.

2.4 Генерация музыкального фрагмента

Для генерации музыкального фрагмента сначала определяется начальное состояние цепи Маркова: на основе сгенерированного равномерно распределенного случайного числа u выбирается подходящий звуковой объект из вектора начальных вероятностей. После этого происходит генерация последовательности звуковых объектов заданной длины аналогичным способом (только с использованием соответсвующих строк матрицы переходных вероятностей вместо вектора начальных вероятностей).

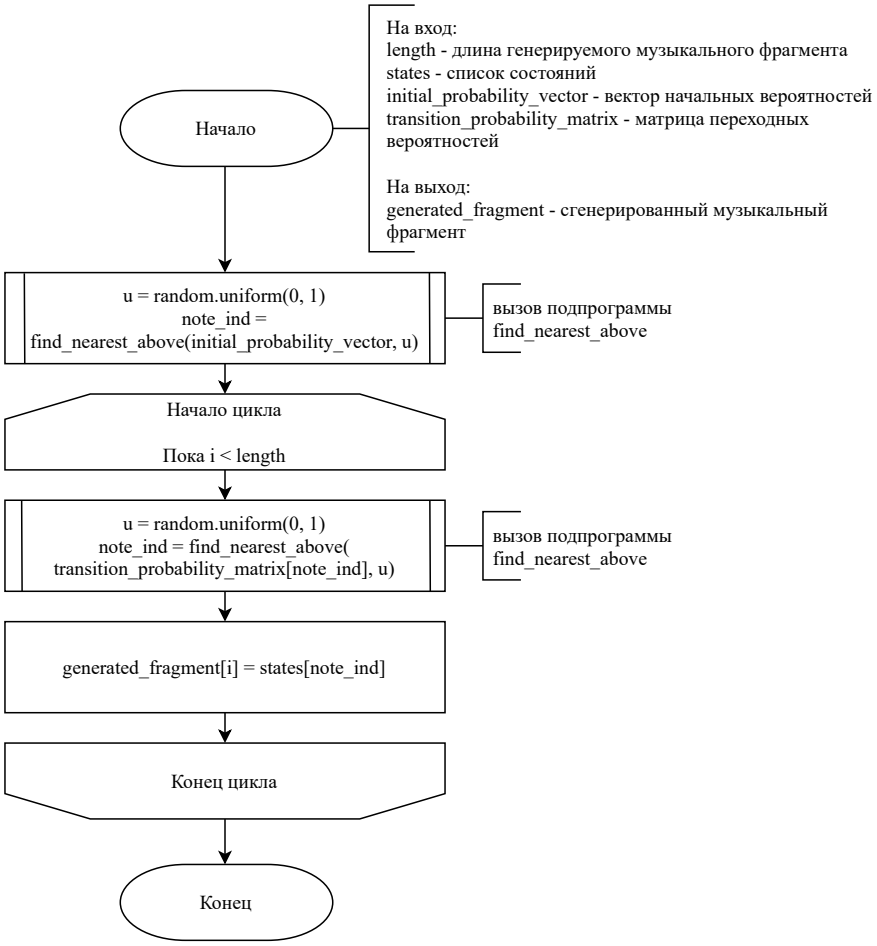

Рисунок 2.7 – Генерация музыкального фрагмента

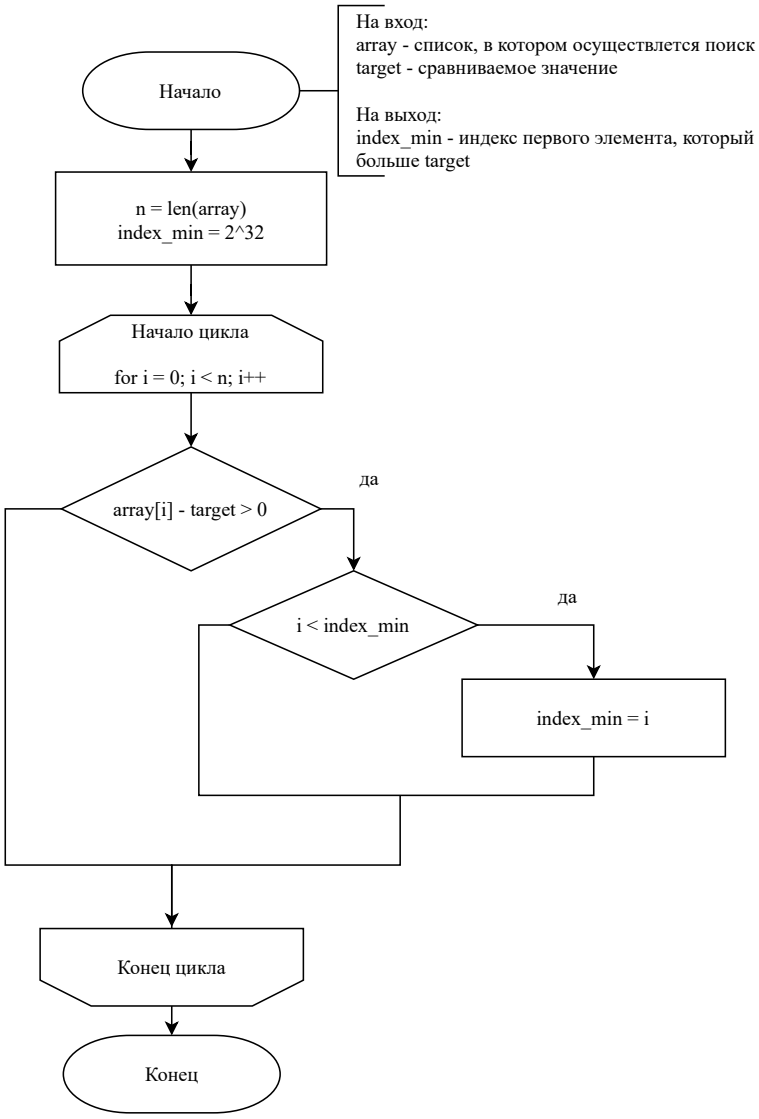

Рисунок 2.8 – Подпрограмма $\text{find}_n earest_a bove$

2.5 Обработка сгенерированного фрагмента

На вход этап получает сгенерированный музыкальный фрагмент, представленный списком звуковых объектов. На выход этап выдает имя MIDI-файла.

Преобразование списка звуковых объектов происходит следующим образом:

1. создание Midi-файла;
2. инициализация переменной offset, которая представляет собой смещение события в Midi-файле;

3. запись в созданный файл мета-события, задающего темп set_tempo;

4. запись элементов списка звуковых объектов в Midi-файл с использованием Midi-событий note_on и note_off;

5. в случае если текущий звуковой объект является нотой, то метке time события note_on присваивается значение offset, а note_off – значение offset + длительность ноты;

6. в случае если текущий звуковой объект является аккордом, то для каждой составляющей его ноты записываются событие note_on с меткой time равной offset и note_off с меткой time равной offset + длительность аккорда;

7. увеличение переменной offset на длительность текущего звукового объекта;

8. сохранение созданного Midi-файла.

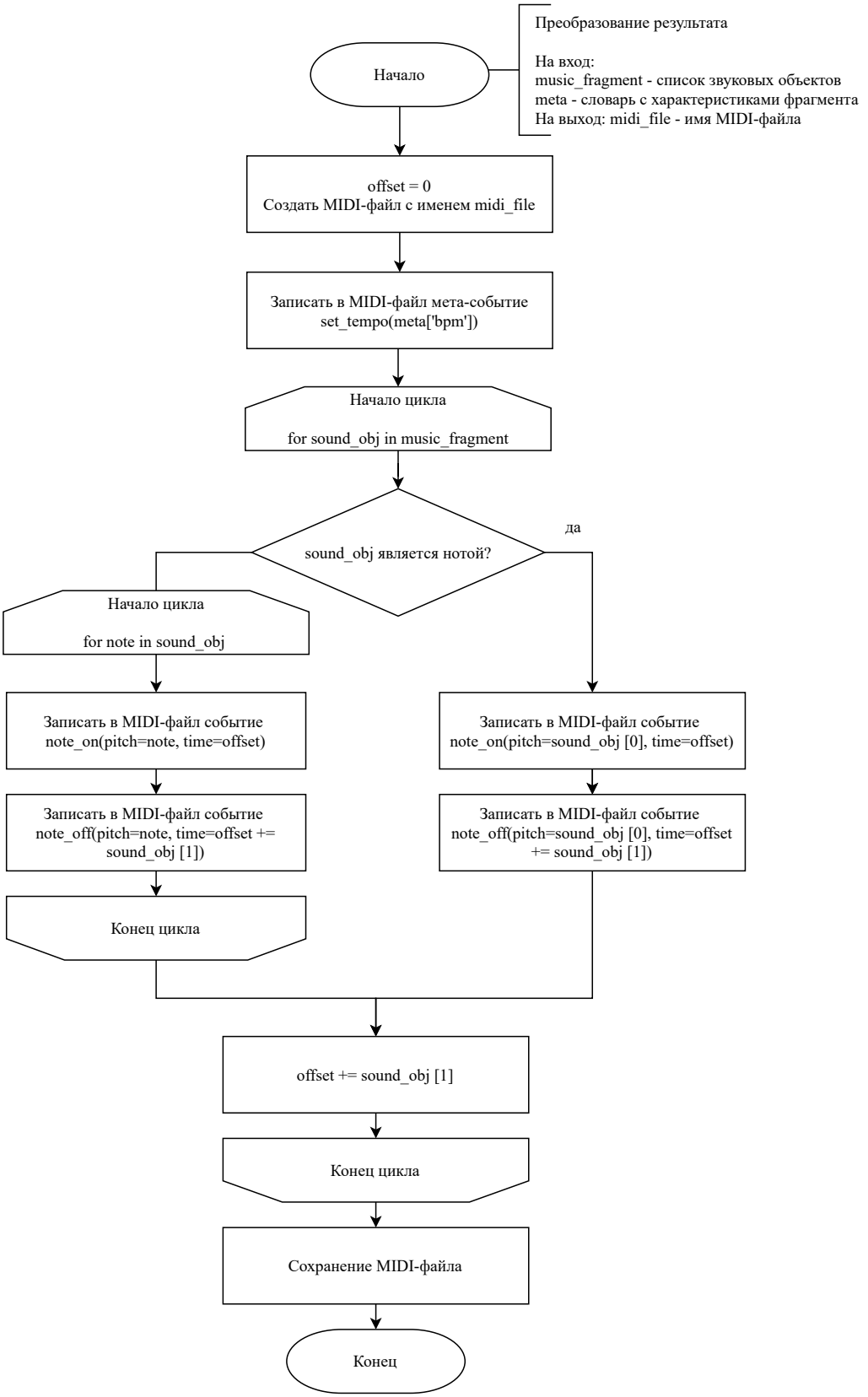

Рисунок 2.9 – Преобразование результата

Вывод

В данном разделе был разработан метод генерации музыкального фрагмента соответствующего эмоциональному состоянию человека с использованием марковских моделей, описаны ключевые этапы в виде схем алгоритмов, а также определены ограничения предметной области.

3 Технологический раздел

3.1 Средства реализации

При написании программного продукта был использован язык программирования Python [18], даныый выбор обусловлен следующими факторами:

— возможность запуска программного кода на любом устройстве с установленным заранее интерпретатором;

— наличие обширного количества актуализируемой справочной литературы

— наличие большого числа библиотек для работы с MIDI-файлами;

— данный язык ориентирован на анализ данных и содержит большое число библиотек для глубокого обучения, что является важным для создания обучающего набора данных.

При написании программного продукта использовалась среда разработки PyCharm 2019 [19], так как данная среда:

— создана компанией JetBrains [20] специально для разработки на языке программирования Python;

— имеет бесплатную версию PyCharm Community Edition, которая распространяется под лицензией Apache License;

— предоставляет все инструменты для написания, отладки и тестирования кода.

В качестве Key-Value хранилища был выбран Redis [21], данный выбор обусловлен следующими факторами:

— Redis хранит записи прямо в оперативной памяти в виде пар ключ-значение, вследствие чего операции чтения/записи в память происходят намного быстрее, чем при работе с дисками;

— наличие клиентской библиотеки Python redis.py [22], которая позволяет вызывать команды Redis из Python и возвращать словари (dict) Python.

3.1.1 Выбор библиотеки для работы с MIDI-файлами

Существует несколько библиотек Python для работы с MIDI-файлами:

1. Mido [23] – это библиотека Python для работы с MIDI-файлами и устройствами ввода/вывода MIDI.
2. Pygame [24] – это библиотека Python для создания игр и мультимедийных приложений. Она также может использоваться для воспроизведения звуковых файлов, включая MIDI-файлы.
3. MIDIUtil [25] – это библиотека Python для создания новых MIDI-файлов.
4. miditoolkit [26] – это библиотека Python для работы с MIDI-файлами.
5. Music21 [27] – это библиотека для анализа, создания и манипулирования музыкальных данных в языке программирования Python.

Ниже, на рисунке 3.1, приведено сравнение существующих библиотек для работы с MIDI-файлами по следующим критериям:

— возможность создания новых MIDI-файлов;

— возможность чтения/записи новых MIDI-файлов;

— возможность воспроизведения MIDI-файлов;

— наличие инструментов для анализа музыкальных данных.

Модуль / Критерий	Mido	Pygame	MIDIUtil	miditoolkit	Music21
Создание новых MIDI-файлов	+	-	+	-	+
Чтение и запись MIDI-файлов	+	-	-	+	+
Воспроизведение MIDI-файлов	-	+	-	-	-
Наличие инструментов для анализа музыкальных данных	-	-	-	-	+

Рисунок 3.1 – Сравнение библиотек для работы с MIDI-файлами

Модуль miditoolkit хорошо подходит для работаты с устройствами ввода/вывода MIDI, Pygame можно использовать, если нужно только воспроизводить звуковые файлы. Если нужно чтение, запись и изменение MIDI-файлов, то для этого подойдет Mido. В случае, если необходимо работать с музыкальными данными и создавать новые MIDI-файлы, то модули Music21 и MIDIUtil могут быть лучшими выборами.

В данной работе будет использоваться модуль Music21.

3.2 Описание входных и выходных данных

Входными данными являются:

— Результат теста Люшера – упорядоченный в порядке препочтения пользователя список цветов.

— Длительность фрагмента – целое число от 25 до 250, представляет собой количество тактов в сгенерированом фрагменте. Количество тактов преобразуется в количество звуковых объектов, которые необходимо сгенерировать, следующим образом: так как музыкальный размер для

всех фрагментов равен $\frac{4}{4}$, то на один такт приходится 4 звуковых объекта, значит:

$$\text{Количество звуковых объектов} = \text{Количество тактов} \cdot 4 \qquad (3.1)$$

Выходными данными является Midi-файл со сгенерированным фрагментом, также предусматривается возможность конвертации фрагмента в формат mp3 для облегчения его дальнейшего распространения и воспроизведения.

3.3 Сведения о модулях

Программное обеспечение состоит из модулей:

— создания обучающего набора данных, модуль использует библиотеку Music21 [27];

— обработки входных данных;

— генерации музыкального фрагмента;

— преобразования выходных данных.

Также определен модуль, который позволяет хранить (обновлять) и получать обучающий набор данных из хранилища Redis.

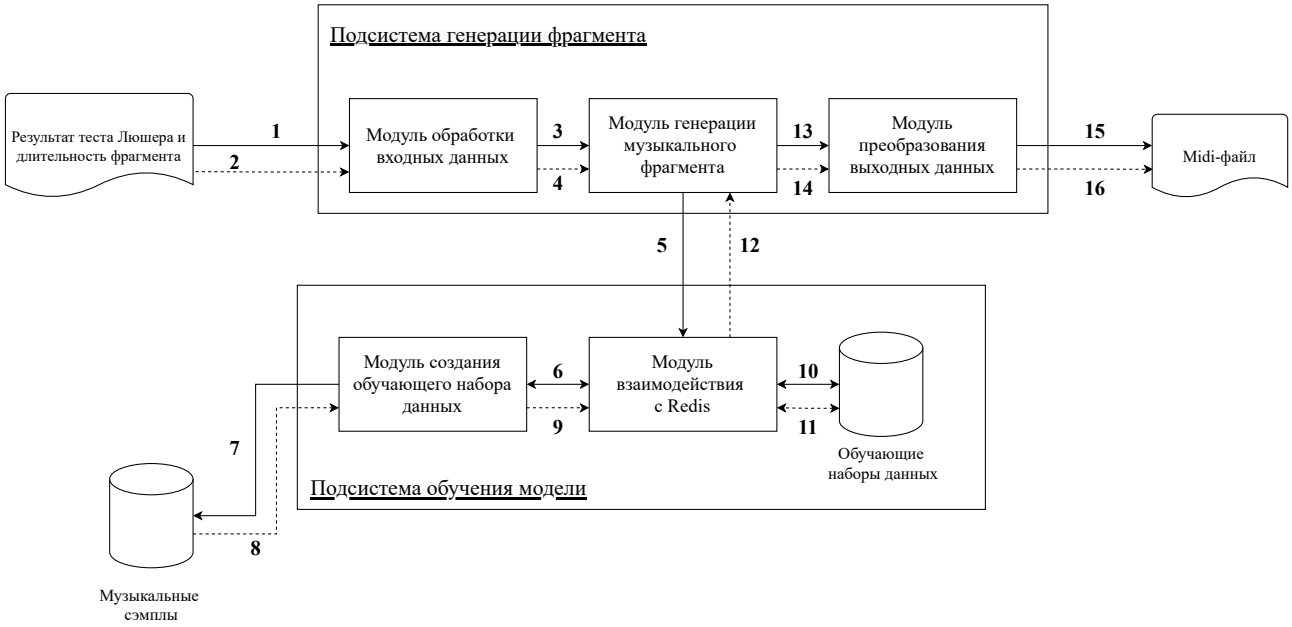

Рисунок 3.2 – Структура программного обеспечения

На рисунке 3.2:

1 – вызов модуля обработки входных данных;

2 – результат теста Люшера и длительность фрагмента;

3 – вызов модуля генерации музыкального фрагмента;

4 – характеристики генерируемого фрагмента;

5 – вызов модуля взаимодействия с Redis;

6 – вызов модуля создания обучающего набора данных;

7 – обращение к хранилищу с музыкальными сэмплами;

8 – музыкальные сэмплы в формате Midi;

9 – обученная модель;

10 – вызов модуля взаимодействия с Redis;

11 – сериализованная обученная модель;

12 – десериализованная обученная модель;

13 – вызов модуля преобразования выходных данных;

14 – музыкальный фрагмент в виде списка звуковых объектов;

15 – обращение к директории сохранения музыкального фрагмента;

16 – музыкальный фрагмент в виде файла в формате Midi.

.

3.4 Описание работы с интерфейсом

На рисунке 3.3 представлен интерфейс разработанного программного обеспечения. В левой части экрана располагается панель управления. В разделе «Информация о сгенерированном фрагменте» будет выводится название обучающего набора данных, использовавшегося для генерации фрагмента, его эмоциональная окраска и длительность в тактах.

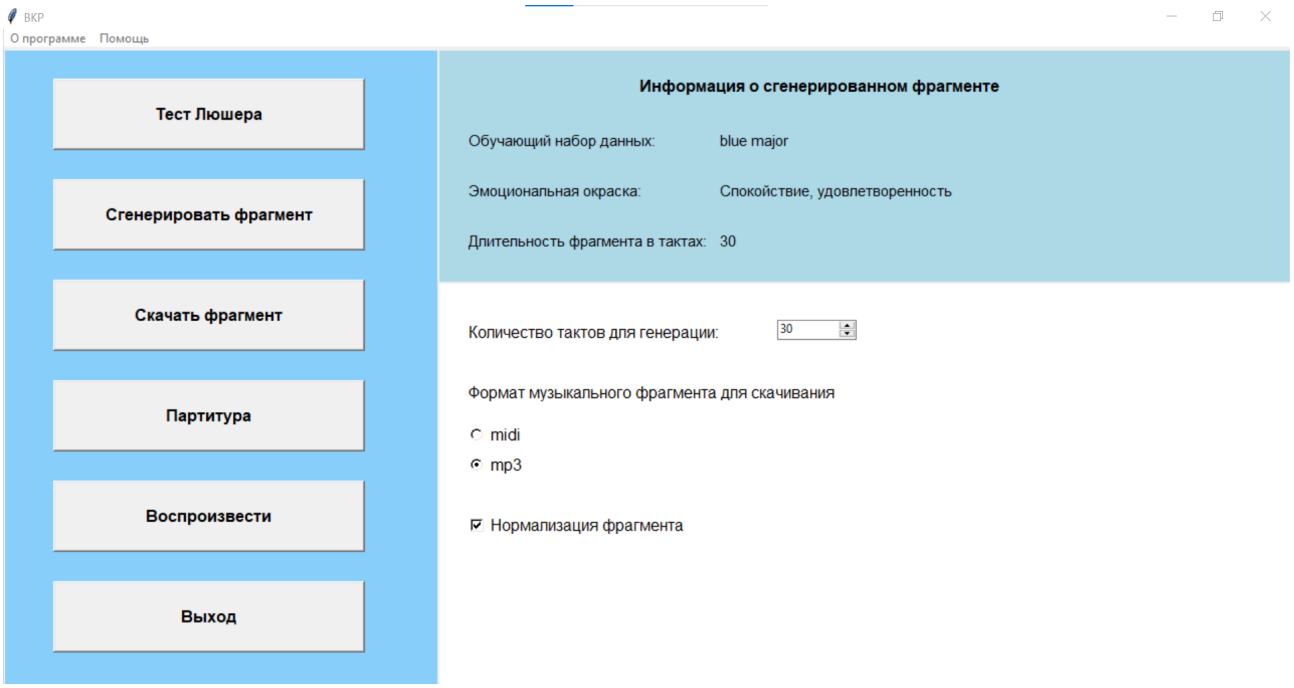

Рисунок 3.3 – Интерфейс ПО

Для генерации музыкального фрагмента необходимо пройти тест Люшера, кликнув на соответсвующую кнопку. Пользователю предлагается выбрать наиболее приятный цвет из представленных, после чего оставшиеся цвета перемешаются и тест повторится пока цвета список цветов не станет пуст.

Рисунок 3.4 – Тест Люшера (первый раунд)

Рисунок 3.5 – Тест Люшера (второй раунд)

После прохождени теста Люшера пользователю предлагается задать длительность генерируемого фрагмента в тактах. В случае если это поле оставить пустым, будет задана длительность по умолчанию, равная 25 тактам. Нажав на кнопку «Сгенерировать фрагмент» можно сгенерировать музыкальный фрагмент, после чего можно:

— Посмотреть нотную партитуру сгенерированного фрагмента 3.6 при помощи кнопки «Партитура».

— Воспроизвести фрагмент при помощи кнопки «Воспроизвести» (при повторном нажатии на кнопку воспроизведение будет прекращено).

— Скачать сгенерированный фрагмент в нужном формате – midi или mp3. В случае скачивания фрагмента в формате mp3 его можно также нормализовать, в результате чего будет изменен уровень громкости аудиофайла так, чтобы он звучал одинаково громко на всех устройствах и в различных условиях воспроизведения. Это позволит избежать скачков громкости и сохранить качество звука. Данное действие необходимо для корректного распространения и воспроизведения сгененрированного фрагмента, так как теперь он сможет быть проигран на различных устройствах без резких перепадов громкости. К Midi-файлам нормализация не применяется, так как они содержат информацию о нотах, инструментах и темпе, но не имеют уровня громкости.

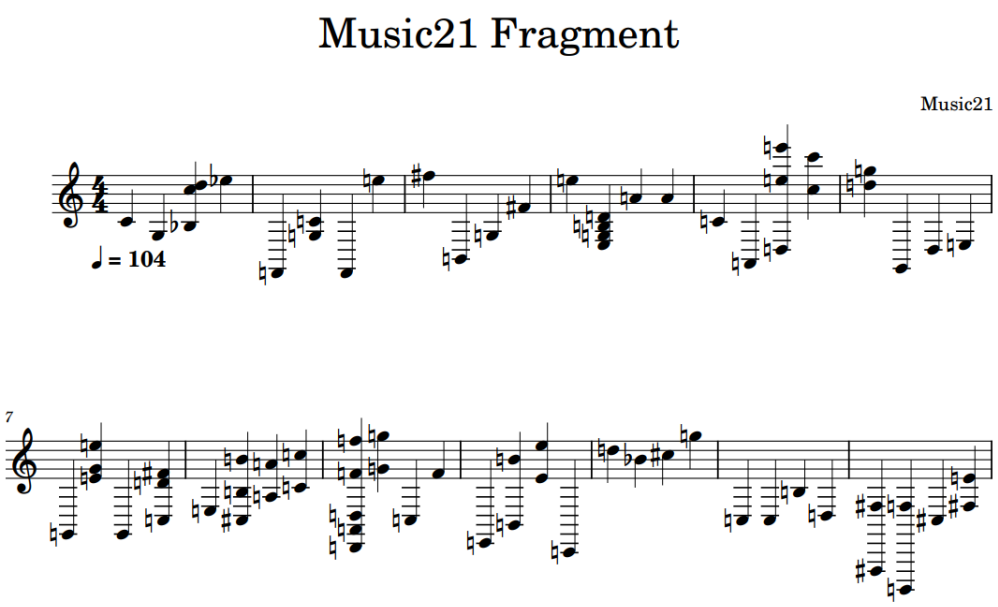

Рисунок 3.6 – Нотная партитура сгенерированного фрагмента

Вывод

В данном разделе был обоснован выбор средств программной разработки метода метода генерации музыкального фрагмента, соответствующего эмоциональному состоянию человека, с использованием марковских моделей и средств для работы с Midi-файлами и хранения обучающего набора данных. Также была описана структура проекта и приведено краткое описание работы пользователя с интерфейсом.

4 Исследовательский раздел

4.1 Планирование исследования

Цель – провести исследование соответствия сгенерированных фрагментов заранее сформированной оценке.

Входные данные: 8 музыкальных фрагментов, сгенерированных с использованием разработанного метода.

Выходные данные: экспертные оценки качества генерируемых фрагментов.

Описание исследования: предполагается взять модель, обученную на 150 произведениях для фортепиано в формате MIDI, по ней получить 8 музыкальных фрагментов и отправить эти фрагменты экспертам для оценки выделенных критериев.

В качестве экспертов были привлечены 72 человека:

— 48 любителей, не имеющих музыкального образования;

— 17 любителей, имеющих музыкальное образование;

— 6 профессионалов в области музыки.

Для того, чтобы эксперты смогли оценить работу метода без временных затрат на установку программного обеспечения, все тестовые фрагменты были переведены в формат mp3, нормализованы и выложены на YouTube, ссылки на соответсвующие видео были приложены в Google форму, которая будет отправлена экспертам по электронной почте.

4.2 Критерии оценки

Для проведения экспертной оценки выделены следующие критерии, которые оцениваются по пятибалльной шкале, где 1 – минимальное значение

критерия, 5 – максимальное:

1. Соответствие сгенерированного фрагмента заданной эмоциональной окраске (Критерий №1):
   - 1 – полностью не соответствует заданной эмоциональной окраске;
   - 2 – скорее не соответствует заданной эмоциональной окраске;
   - 3 – более-менее соответствует заданной эмоциональной окраске;
   - 4 – скорее соответствует заданной эмоциональной окраске;
   - 5 – соответствует заданной эмоциональной окраске.
2. Реалистичность звучания инструмента (фортепиано) (Критерий №2):
   - 1 – инструмент звучит искусственно;
   - 2 – инструмент скорее звучит искусственно;
   - 3 – инструмент звучит отчасти искусственно, отчасти реалистично;
   - 4 – инструмент скорее звучит реалистично;
   - 5 – инструмент звучит реалистично.
3. Приятность/благозвучность мелодии для восприятия (не по структуре, а по отдельным фразам трека) (Критерий №3):
   - 1 – фрагмент звучит не связано, он представляет собой набор отдельных нот/аккордов;
   - 2 – фрагмент звучит скорее не связано, некоторые сочетания нот/аккордов подходят, но в основном он представляет собой набор отдельных нот/аккордов;
   - 3 – фрагмент звучит более-менее связано, некоторые отрывки фрагмента подходят друг к другу, но он содержит много несвязанных участков;
   - 4 – фрагмент звучит скорее благозвучно, в нем прослеживается паттерн, но некоторые ноты/аккорды не попадают в такт или выбиваются из гаммы;

— 5 – фрагмент звучит полностью благозвучно, ноты/аккорды сочетаются и попадают в гамму.

4. Цельность фрагмента (Критерий №4):

   — 1 – фрагмент звучит абсолютно не цельно;

   — 2 – фрагмент звучит скорее не цельно, однако прослеживаются некоторые связанные части;

   — 3 – фрагмент звучит более-менее цельно, частично попадаются связанные части;

   — 4 – фрагмент звучит скорее цельно, однако попадаются несвязанные ноты/аккорды;

   — 5 – фрагмент звучит полностью цельно.

5. Реалистичность отдельных фраз фрагмента (критерием реалистичности является вероятность создании данного фрагмента человеком. Динамика при записи фрагмента не учитывалась) (Критерий №5):

   — 1 – фрагмент звучит абсолютно нереалистично;

   — 2 – фрагмент звучит скорее не реалистично;

   — 3 – фрагмент звучит более-менее реалистично;

   — 4 – фрагмент звучит скорее реалистично;

   — 5 – фрагмент звучит полностью реалистично, как будто его написал композитор.

4.3 Обработка результатов опроса

Для каждого из m раундов опроса составляется таблица 4.5. Элементы таблицы, величины x_{jk}, получены в результате опроса экспертов и представляют собой баллы, выставленные по пятибальной шкале.

Таблица 4.1 – Результат проведения i-го раунда опроса

	Кр.1	Кр.2	Кр.3	Кр.4	Кр.5
Эксперт №1	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}
Эксперт №2	x_{21}	x_{22}	x_{23}	x_{24}	x_{25}
...	...	...	...	...	...
Эксперт №n	x_{n1}	x_{n2}	x_{n3}	x_{n4}	x_{n5}

Далее вычисляется обобщенная экспертная оценка критериев для каждого раунда опроса и сводится в таблицу 4.2.

Элемент Q_{ij} таблицы 4.2 является обобщенной (усредненной) экспертной оценкой j-го критерия в i-ом раунде опроса и вычисляется как:

$$Q_{ij} = \frac{1}{n} \sum_{k=1}^{n} x_{jk}^{i}, \tag{4.1}$$

где n – количество экспертов, x_{jk}^{i} – оценка в баллах, выставленная k-м экспертом j-му критерию в i-м раунде опроса.

Таблица 4.2 – Результат экспертной оценки для каждого раунда

	Кр.1	Кр.2	Кр.3	Кр.4	Кр.5
Раунд №1	Q_{11}	Q_{12}	Q_{13}	Q_{14}	Q_{15}
Раунд №2	Q_{21}	Q_{22}	Q_{23}	Q_{24}	Q_{25}
...	...	...	...	...	...
Раунд №m	Q_{m1}	Q_{m2}	Q_{m3}	Q_{m4}	Q_{m5}

Далее вычисленные значения сводятся в результирующую таблицу 4.3.

Элемент Q_j таблицы 4.3 является обобщенной (усредненная) экспертная

оценкой j-го критерия и вычисляется как:

$$Q_j = \frac{1}{m} \sum_{i=1}^{m} Q_{ji}, \tag{4.2}$$

где m – количество опытов, Q_{jl} – усредненная оценка j-го критерия в i-ом раунде опроса.

Таблица 4.3 – Результат экспертной оценки

	Среднее значение по всем раундам
Критерий №1	Q_1
Критерий №2	Q_2
...	...
Критерий №k	Q_k

4.4 Результат исследования

Программа, реализующая метод генерации музыкального фрагмента, соответсвующего эмоциональному состоянию человека, с использованием марковских моделей, была обучена на 150 произведениях для фортепиано. После обучения был сгенерирован набор из 8 музыкальных фрагментов (по одному фрагменту на обучающий набор). Данные фрагменты были отправлены на анализ 72 экспертам, которые должны были оценить каждое произведение по выделенным ранее пяти критериям. Данные от каждого эксперта были обработаны, сведены в таблицу 4.5 и представлены на графиках 4.1 - 4.2.

На графике 4.1 провизуализировано, какие оценки чаще всего выставлялись каждому из критериев.

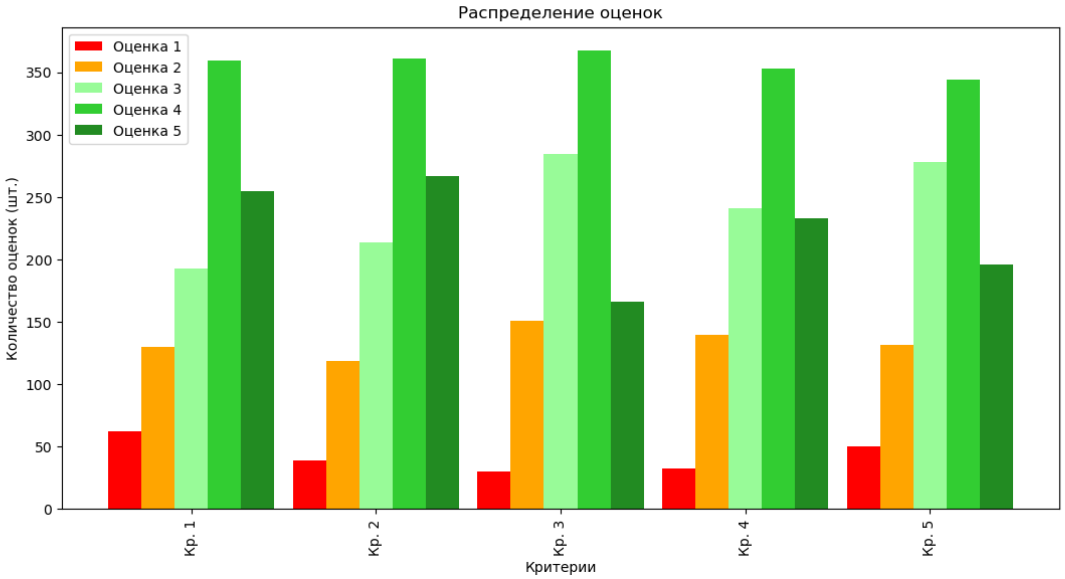

Рисунок 4.1 – Распределение оценок

На графике 4.2 продемонстрировано распределение оценок по квартилям:

— Q1 – нижний (первый) квартиль, 25% значений выборки меньше Q1.

— Q2 – второй квартиль, медиана выборки.

— Q3 – верхний (третий) квартиль, 25% значений выборки больше Q3.

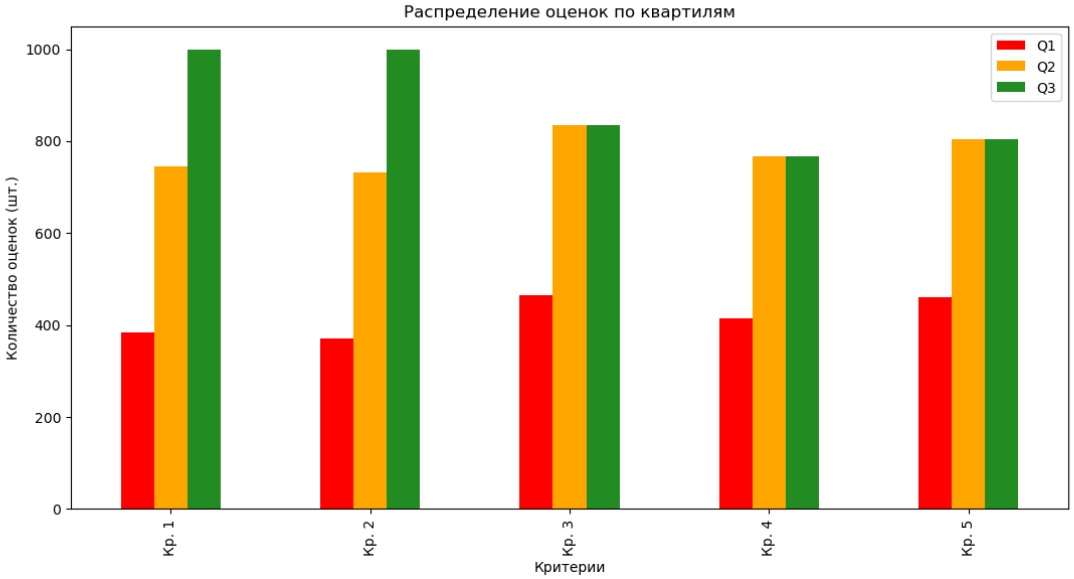

Рисунок 4.2 – Распределение оценок по квартилям

Исходя из графика 4.2, медиана для всех критериев равна 4 баллам, что является достаточно хорошим показателем для сгенерированной компьютером

музыки. На графике 4.1 можно увидеть, что всем критериям также чаще всего выставлялись оценки 4.

Ниже, в таблице 4.2, приведены усредненные оценки критериев по всем раундам опроса.

Таблица 4.4 – Средние оценки для каждого критерия по всем раундам опроса

	Среднее значение по всем раундам
Критерий №1	3,62
Критерий №2	3,7
Критерий №3	3,49
Критерий №4	3,61
Критерий №5	3,50

Делая вывод по каждому критерию, можно сказать, что критерии №3 «Приятность/благозвучность мелодии для восприятия» и №5 «Реалистичность отдельных фраз фрагмента» имеют самые низкие средние оценки, что является естественным результатом для компьютерной генерации музыки. По мнению экспертов в сгенерированных фрагментах не хватает динамики, что обуслевлено фактом, что плотность звучания достигается за счет уплотнения музыкального текста. Улучшение критериев №3 и №5 повлечет за собой улучшение качества сгенерированных фрагментов. Это может быть достигнуто использованием марковских моделей более высоких порядков для алгоритма генерации.

Наиболее важный критерий для цели исследования – «Соответствие сгенерированного фрагмента заданной эмоциональной окраске» был оценен экспертами достаточно высоко: чаще всего эксперты оценивали его в 4 и 5 баллов, а его среднее значение по всем раундам состовляет 3.62 балла. Для повышения оценки данного критерия необходимо улучшить способ определения эмоциональной окраски сэмлов из обучающей выборки.

Критерий №2 «Реалистичность звучания инструмента» в среднем был оценен на 3.7 балла, чаще всего ему выставлялись оценки 3 и 4, из чего можно сделать вывод, что не совсем корректно был выбран плагин, переводящий сгенерированный фрагмент из формата .mid в формат .mp3.

Критерий №4 «Цельность фрагмента» был в среднем оценен на 3.61 балла, чаще всего ему выставлялась оценка 4. Улучшение этого критерия возможно за счет увеличения размера обучающей выборки, а также повышения ее однородности.

4.4.1 Оценка согласованности

Для оценки степени согласованности мнений экспертов был рассчитан коэффициент вариации (CV) экспертной оценки, который является мерой разброса данных и определяется как отношение стандартного отклонения к среднему значению. Он показывает, насколько велик разброс экспертных оценок относительно их среднего значения. Чем меньше значение коэффициента вариации, тем более точными и согласованными считаются экспертные оценки.

Формула для расчета коэффициента вариации:

$$CV = \frac{\sigma}{\overline{x}} \cdot 100\%. \tag{4.3}$$

Стандартное отклонение:

$$\sigma = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(x_i - \overline{x})}, \tag{4.4}$$

среднее значение экспертной оценки:

$$\overline{x} = \frac{1}{n}\Sigma_{i=1}^{n}x_i, \tag{4.5}$$

где n – количество выставленных оценок, x_i – i-я выставленная оценка.

Таблица 4.5 – Коэффициент вариации для критериев оценки

	Коэффициент вариации
Критерий №1	38%
Критерий №2	32%
Критерий №3	30%
Критерий №4	32%
Критерий №5	34%

Для критерия «Соответствие сгенерированного фрагмента заданной эмоциональной окраске» значение коэффициента составило 38%, что является допустимым значением. Для всех остальных критериев коэффициент вариации находится в диапазоне от 30 до 34%, что говорит о согласованности экспертных оценок.

Вывод

В целом можно сказать, что общее впечатление от сгенерированных музыкальных фрагментов у экспертов осталось положительное. Среди минусов эксперты выделяют недостаточную реалистичность и нехватку динамики сгенерированных фрагментов, а также недостаточную реалистичность звучания инструмента (фортепиано).

Заключение

В ходе работы были проанализированы существующие методы алгоритмической композиции, проведен сравнительный анализ методов алгоритмической композиции и выбран метод наиболее подходящий для решаемой задачи. Разработан метод генерации музыкального фрагмента, соответсвующего эмоциональному состоянию человека, и программное обеспечение, реализующее данный метод. Также была проведена экспертная оценка разработанного метода.

В рамках работы решены все поставленные задачи:

— Проведен анализ предметной области генерации музыкальной композиции и существующих методов алгоритмической композиции.

— Разработан метод генерации музыкального фрагмента, соответсвующего эмоциональному состоянию человека, с использованием марковских моделей.

— Разработано программное обеспечение, реализующее спроектированный метод.

— Проведено исследование соответствия сгенерированных фрагментов заранее сформированной оценке методом экспертной оценки.

Дальнейшее развитие проекта может быть направлено на улучшение способа генерации музыкального фрагмента, так как на данный момент ноты и аккорды сгенерированного фрагмента не всегда могут находиться в соответствии, в следствии чего могут возникать диссонирующие звуки. Также на данном этапе в сгенерированном музыкальном фрагменте не хватает не хватает динамики, так как плотность звучания достигается за счет уплотнения музыкального текста. Одним из решений может стать использование марковских моделей более высокого порядка или скрытых марковских моделей.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Inc M.* Mubert [Электронный ресурс]. — Режим доступа: `https://mubert.com/`. — (Дата обращения: 06.11.2022).
2. LMUSe [Электронный ресурс]. — Режим доступа: `https://www.oocities.org/hacad/lmuse/lmuse.html`. — (Дата обращения: 06.11.2022).
3. *Liang F.* BachBot: Automatic composition in the style of Bach chorales– Developing, analyzing, and evaluating a deep LSTM model for musical style. — 2016.
4. *Kim J. S.* DeepJazz [Электронный ресурс]. — Режим доступа: `https://deepjazz.io/`. — (Дата обращения: 06.11.2022).
5. *AI G.* NSynth [Электронный ресурс]. — Режим доступа: `https://magenta.tensorflow.org/nsynth`. — (Дата обращения: 06.11.2022).
6. *Bozhanov B.* Computoser [Электронный ресурс]. — Режим доступа: `http://computoser.com/`. — (Дата обращения: 06.11.2022).
7. *Bernardini M.* Meldy [Электронный ресурс]. — Режим доступа: `https://github.com/bubblefishstudio/meldy.`. — (Дата обращения: 06.11.2022).
8. *Nierhaus G.* Algorithmic composition: paradigms of automated music generation. — 2009.
9. *Никитин Н.А., Розалиев В.Л., Орлова Ю.А.* Обзор математических методов для генерации музыкальных композиций // Молодой ученый Международный научный журнал №52, Выпуск №342. — 2020. — С. 36—39.
10. *Славщик А.* Эволюционные методы в алгоритмической композиции. л-системы. — 2013.
11. *McCormack J.* Grammar based music composition // Complex systems, Volume №96. — 1996. — С. 321—336.
12. *K. Saravanan, S. Sasithra.* Review on Classification Based on Artificial Neural Networks // The International Journal of Ambient Systems and Applications, Volume №2. — 2014. — Дек. — С. 11—18.
13. *AI G.* Magenta [Электронный ресурс]. — Режим доступа: `https://magenta.tensorflow.org/`. — (Дата обращения: 06.11.2022).

14. *Lüscher M.* The Luscher color test. — Simon, Schuster, 1990.

15. *Coronel C.*, *Morris S.* Database systems : design, implementation, and management. — Cengage Learning, 2016.

16. *Sullivan D.* NoSQL for mere mortals. — Addison-Wesley Professional, 2015.

17. *Perkins L.*, *Wilson J.*, *Redmond E.* Seven databases in seven weeks: a guide to modern databases and the NoSQL movement // Seven Databases in Seven Weeks. — 2018. — С. 1—325.

18. *Foundation P. S.* Python ORG [Электронный ресурс]. — Режим доступа: `https://docs.python.org/3/index.html`. — (Дата обращения: 09.04.2023).

19. *JetBrains.* PyCharm [Электронный ресурс]. — Режим доступа: `https://www.jetbrains.com/pycharm/`. — (Дата обращения: 09.04.2023).

20. *s.r.o J.* JetBrains [Электронный ресурс]. — Режим доступа: `https://www.jetbrains.com/`. — (Дата обращения: 01.05.2023).

21. *Ltd R.* Redis [Электронный ресурс]. — Режим доступа: `https://redis.io/`. — (Дата обращения: 09.04.2023).

22. *Ltd R.* Python with Redis [Электронный ресурс]. — Режим доступа: `https://developer.redis.com/develop/python/`. — (Дата обращения: 01.05.2023).

23. *Association M.* Midi ORG [Электронный ресурс]. — Режим доступа: `https://midi.org`. — (Дата обращения: 04.03.2023).

24. *Foundation P. S.* Pygame [Электронный ресурс]. — Режим доступа: `https://pypi.org/project/pygame/`. — (Дата обращения: 09.05.2023).

25. *Foundation P. S.* MIDIUtil [Электронный ресурс]. — Режим доступа: `https://pypi.org/project/MIDIUtil/`. — (Дата обращения: 09.05.2023).

26. *Foundation P. S.* miditoolkit [Электронный ресурс]. — Режим доступа: `https://pypi.org/project/miditoolkit/`. — (Дата обращения: 09.05.2023).

27. *Foundation P. S.* Music21 [Электронный ресурс]. — Режим доступа: `https://pypi.org/project/music21/`. — (Дата обращения: 01.05.2023).

Приложение А