

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МО и ПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

**Разработка приложения для генерации
матрицы серийных форм**

Студент _____ Валковой Владислав Викторович
(подпись, дата) (Ф. И. О., полностью)

Руководитель _____ Кольчугина Е. А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д. В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 03.06.2026 г. № 12)

Заведующий кафедрой _____ Козлов А. Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н. А.
(подпись) (фамилия, инициалы)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МО и ПЭВМ

«УТВЕРЖДАЮ»
Заведующий кафедрой
_____ А. Ю. Козлов
« 26 » _____ декабря 2025 г.

**ЗАДАНИЕ НА
ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент _____ Валковой Владислав Викторович _____ гр. _____ 22ВИ1 _____ факультета _____ ВТ _____
(Ф. И. О., полностью)

2. Тема работы _____ Разработка приложения для генерации матрицы серийных форм _____

Тема утверждена приказом ПГУ № _____ 1583/о _____ от « 26 » _____ декабря 2025 г.

3. Руководитель работы _____ профессор кафедры «МО и ПЭВМ» Кольчугина Е. А. _____

4. Задание на работу (название разработки, исходные данные и т. п.):

1. Основное требование: разработка консольного приложения для генерации матрицы серийных форм, которое должно строить матрицу форм предложенной двенадцатитоновой серии.

2. Требования к среде функционирования (аппаратно-программные требования): программа должна функционировать как консольное приложение под управлением операционной системы Windows, версии не ниже 7.

3. Требования к среде разработки: язык программирования – C++; библиотеки – стандартные и предлагаемые пакетами Microsoft Visual Studio.

4. Основные функции разрабатываемой программы:

а) ввод двенадцатитоновой серии;

б) редактирование серии;

в) вывод серии на экран;

г) анализ качества серии;

д) построение матрицы форм серии;

е) сохранение серии в файл;

ж) сохранение матрицы в файл.

5. Перечень подлежащих разработке вопросов:

- 1) Анализ предметной области и постановка задачи;
- 2) Проектирование;
- 3) Реализация;
- 4) Контроль качества программного обеспечения;
- 5) План разработки бюджета;
- 6) Оформление пояснительной записки.

6. Календарный график выполнения работы:

№ п/п	Наименование этапов работы	Объём работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10 %	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20 %	14.04.2026 – 25.05.2026	
3	Реализация программных средств	30 %	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20 %	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20 %	22.05.2026 – 02.06.2026	

Дата выдачи задания: « 26 » декабря 2025 г.

Руководитель бакалаврской работы _____ 26.12.2025 г.
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025 г.
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 157 листов, 62 рисунка, 10 таблиц, 31 листинг, 35 использованных источников, 5 приложений.

СЕРИЯ, СЕРИЙНЫЕ ФОРМЫ, МАТРИЦА СЕРИЙНЫХ ФОРМ, ДОДЕКАФОНΙΑ, КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ WINDOWS, C++, MICROSOFT VISUAL STUDIO

Объектом разработки является приложение для генерации матрицы серийных форм. Целью курсового проекта является разработка информационной системы анализа магазина фурнитуры, выполняющей хранение и обработку данных фурнитуры.

Цель выпускной квалификационной работы – разработка приложения для генерации матрицы серийных форм, которая позволит упростить процесс анализа серии и построения её форм.

Технология разработки – консольное приложение Windows на языке программирования C++. Логика приложения включает обработку поступающих от пользователя команд и многошаговые действия. Среда разработки – Microsoft Visual Studio 2022.

В результате выпускной квалификационной работы было разработано приложение для генерации матрицы серийных форм. Все требования к разработке приложения соблюдены.

					ПГУ 090302 – 08ВР221.08 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Разработка приложения для генерации матрицы серийных форм Пояснительная записка	Лит.	Лист	Листов
Разраб.		Валковой В.В.					4	157
Провер.		Кольчугина Е.А.						
Реценз.								
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.				группа 22ВИ1		

Оглавление

Введение.....	7
1 Специфика предметной области.....	8
1.1 Обзор традиционных графических средств нотописи.....	8
1.2 Тон. Строй. Звукоряд.....	10
1.2.1 Понятие тона.....	10
1.2.2 Понятие строя и звукоряда.....	11
1.3 Знаки альтерации.....	12
1.4 Интервалы.....	13
1.5 Октавная система.....	15
1.6 Серия и матрица её форм.....	17
1.6.1 Серия. Формы серии.....	17
1.6.2 Матрица серийных форм.....	19
2 Постановка задачи и анализ требований.....	22
2.1 Постановка задачи.....	22
2.2 Анализ систем аналогичного назначения.....	23
2.2.1 Анализ программы Sibelius.....	23
2.2.2 Анализ программы MuseScore.....	24
2.2.3 Анализ приложения ToneMatrix.....	25
2.2.4 Сравнительный анализ аналогичных систем.....	25
2.3 Формулировка требований к разрабатываемому приложению ...	27
2.4 Выбор и обоснование технологии.....	32
3 Проектирование приложения.....	33
3.1 Проектирование структуры данных.....	33
3.2 Проектирование алгоритмов.....	37

3.3 Проектирование интерфейса	41
4 Программная реализация.....	47
5 Контроль качества программного обеспечения и тестирование приложения.....	55
5.1 Выбор методов контроля качества и тестирования программного обеспечения.....	55
5.2 Оценка качества кода метриками SLOC	57
5.3 Оценка качества кода метриками Холстеда.....	59
5.4 Функциональное тестирование приложения	61
6 Планирование разработки и оценка бюджета	71
Заключение	73
Список использованных источников	75
Приложение А (справочное) Глоссарий	79
Приложение Б (рекомендуемое) Код программы.....	84
Приложение В (рекомендуемое) Результаты тестирования.....	147
Приложение Г (рекомендуемое) Копия акта о внедрении	154
Приложение Д (рекомендуемое) Диаграмма Ганта	156

Введение

Многие вычислительные процессы, некогда выполняемые вручную, сегодня автоматизированы: начиная с обыкновенных арифметических операций и заканчивая сложными и ресурсозатратными моделированиями взаимодействий атомов. Автоматизированы были и некоторые процедуры в социальной сфере: например, опросы и тестирование. Без особого внимания осталась разве что сфера, связанная с музыкальной педагогической и композиторской практикой. В средних и высших учебных заведениях по сей день проверяются задачи по гармонии, полифонии и прочим теоретическим дисциплинам вручную. На это, как правило, затрачивается приличное количество времени. Как известно, рост производительных сил (соответственно и автоматизация труда) ведёт к сокращению рабочего времени и обеспечению всестороннего развития каждого члена общества [1]. Педагоги и композиторы – не исключение.

Настоящая работа фокусируется на автоматизации одной операции – построение матрицы форм двенадцатитоновой серии. Матрица значительно облегчает процесс анализа и написания произведений додекафонной музыки, однако построение её весьма трудоёмко и монотонно. Проще говоря, эта операция располагает к автоматизации средствами электронно-вычислительной машины.

Перед работой были определены следующие цели:

- а) изучить аналогичные программные продукты и провести их сравнительный анализ;
- б) выполнить проектирование архитектуры приложения и структуры данных разрабатываемого программного обеспечения;
- в) выполнить разработку программного обеспечения;
- г) провести тестирование разработанного приложения;
- д) оформить пояснительную записку в соответствии с требованиями ГОСТ.

1 Специфика предметной области

Данная глава опирается на учебник советского музыковеда, теоретика музыки и педагога Игоря Владимировича Способина [2]. Его труд пользуется большим уважением в академических кругах и сегодня, а для настоящей работы он отлично подойдёт в качестве материала к разьяснению аспектов некоторых элементарных единиц музыкально-теоретического аппарата, без которых восприятие результатов настоящей работы может оказаться затруднительным.

1.1 Обзор традиционных графических средств нотописи

Для обозначения звука принят овал, внутри пустой или зачёркнутый (рисунок 1).



Рисунок 1 – Символы для обозначения звука

Для обозначения длительности звука добавляются ещё некоторые знаки, однако этот аспект предметной области интереса для работы не представляет.

Нотным станом (нотноосцем) называется система из пяти параллельных горизонтальных линий, на которой размещаются ноты. Счёт линий ведётся снизу вверх (рисунок 2).

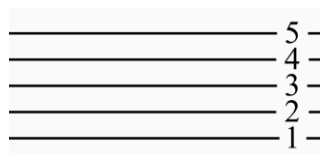
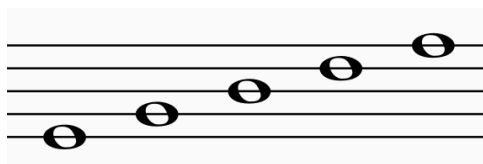
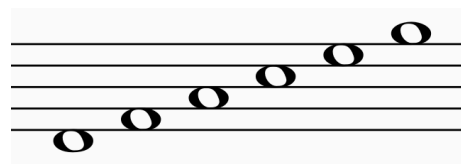


Рисунок 2 – Схема нотного стана

Часть нот размещается на линиях (рисунок 3, а). При этом для того, чтобы определить местонахождение ноты, говорят, что нота находится на 1-й линии, на 2-й линии и так далее. Часть – в промежутках между линиями (то есть над или под ними) (рисунок 3, б).



а)



б)

Рисунок 3 – Схема нотного стана с расположением на нём нот

Для обозначения высоты нот, выходящих за границы нотного стана применяют так называемые добавочные линейки. Это короткие линии для одной ноты, помещаемые выше или ниже нотного стана. Счёт добавочных линий сверху ведётся в восходящем направлении, снизу – в нисходящем (рисунок 4).

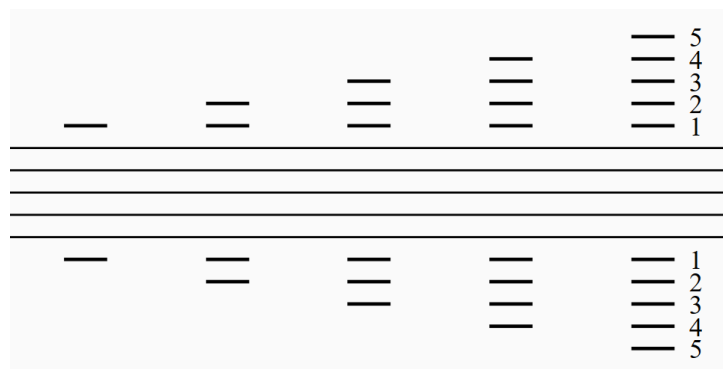


Рисунок 4 – Схема добавочных линеек на нотном стане

В музыкальном письме существуют символы для ассоциации определённой линии стана с конкретной высотой (тоном). Называются они ключами, их три: скрипичный, ключ С (До) и басовый. В рамках разрабатываемого приложения потребуется использовать лишь первый из них.

Скрипичный ключ обозначает g^1 и ставится на 2-й линии нотного стана (рисунок 5).



Рисунок 5 – Схема скрипичного ключа

Этих графических компонентов вполне достаточно для описания двенадцатитоновой серии¹.

1.2 Тон. Строй. Звукоряд

1.2.1 Понятие тона

Тон – одно из основных, широко используемых в теории музыки понятий. Тон имеет высоту и громкость. Отличается от звука музыкального, состоящего из основного тона и обертонов [3].

Для обозначения названия тона используются буквы латинского алфавита. Существуют три системы нотации (таблица 1).

Таблица 1 – Немецкая и американская системы звуковой нотации

№ п/п	Наименование слоговое (по-русски)	Написание в немецкой системе	Написание в американской системе
1	<i>до</i>	<i>c</i>	C
2	<i>ре</i>	<i>d</i>	D
3	<i>ми</i>	<i>e</i>	E
4	<i>фа</i>	<i>f</i>	F
5	<i>соль</i>	<i>g</i>	G
6	<i>ля</i>	<i>a</i>	A
7	<i>си</i>	<i>h</i>	B
8	<i>си^б / си-бемоль</i>	<i>b</i>	B ^b
9	<i>♯ / -диез</i>	<i>-is</i>	♯
10	<i>♭ / -бемоль</i>	<i>-es</i>	♭
11	<i>𝄌 / -дубль-диез</i>	<i>-isis</i>	не употребляется в широкой практике
12	<i>𝄍 / -дубль-бемоль</i>	<i>-eses</i>	не употребляется в широкой практике
13	<i>𝄎 / -бекар</i>	—	—

¹ Подробнее о двенадцатитоновой серии – в главе 1.5.

Как можно заметить, существенным отличием немецкой и американской систем является запись тона *b*: в одном случае – это *си-бемоль*, в другом – просто *си*. В программном обеспечении, предназначенном для работы с единицами музыкально-теоретического аппарата, применяется американская система, так как при этом устраняется необходимость обрабатывать исключение с *b*.

Более того, при работе с немецкой системой придётся работать с исключением, касающимся знаков альтерации *-es* и *-eses*: ко всем тонам, чьё название определяется гласной буквой, бемоль (и дубль-бемоль) записывается без первой буквы: *ми-бемоль* – *es*, *ля-бемоль* – *as*, *ми-дубль-бемоль* – *eses* и *ля-дубль-бемоль* – *ases*.

Однако стоит учесть, что в современной отечественной академической среде музыкальных теоретиков традиционно применяется немецкая система.

1.2.2 Понятие строя и звукоряда

Строем называется совокупность постоянных отношений по высоте между звуками музыкальной системы.

С начала XVIII века в европейской музыке принята двенадцатизвуковая темперация, делящая октаву на двенадцать равных друг другу частей, называемых полутонами. Таким образом, полутон – наименьшее расстояние по высоте в равномерно темперированном строе². Соответственно, целый тон включает два полутона.

Звукорядом называется совокупность звуков музыкальной системы, расположенных в порядке высоты (в восходящем или нисходящем направлении). Между основными ступенями³ звукоряда: пять целых тонов: *c – d*, *d – e*, *f – g*, *g – a*, *a – h*; и два полутона: *e – f*, *h – c* (рисунок 6).

² Равномерно темперированный строй – это темперированный музыкальный строй, в котором каждая октава делится на математически равные интервалы, в наиболее типичном случае на двенадцать полутонов, каждый из которых равен $\sqrt[12]{2}$. Такой строй господствует в европейской профессиональной музыке (академической и эстрадной) начиная с XVIII века до наших дней [4].

³ Ступень – звук музыкальной системы. Обозначается римской цифрой.

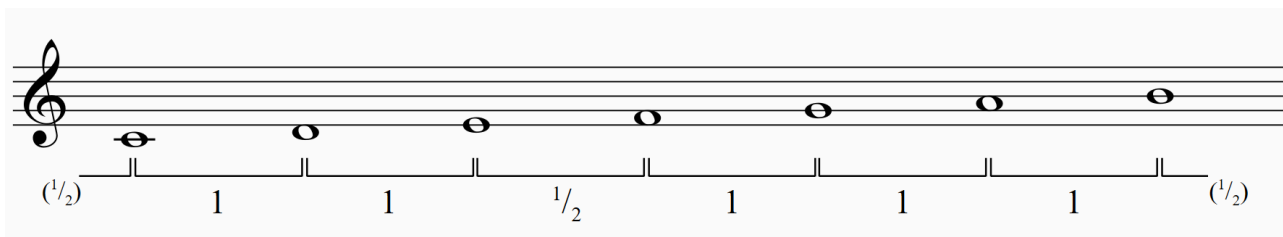


Рисунок 6 – Схема звукоряда с расстояниями между звуками

Помимо натуральных, «белых»⁴ звуков есть обозначения для определения остальных пяти, называемых производными.

1.3 Знаки альтерации

Каждая основная ступень может быть как повышена, так и понижена. Это явление получило название альтерация.

Знаками альтерации называются знаки, ставящиеся слева от ноты и указывающие на повышение или понижение основной ступени, которой соответствует эта нота.

Знаков альтерации пять:

- а) дубль-диез, обозначающий повышение на целый тон;
- б) диез, обозначающий повышение на полутон;
- в) бекар, отменяющий действие предыдущих знаков альтерации;
- г) бемоль, обозначающий понижение на полутон;
- д) дубль-бемоль, обозначающий понижение на целый тон.

Если перед нотой не стоит никакого знака, это означает: либо чистый тон (белая клавиша на фортепиано), либо продолжается действие предыдущего знака альтерации⁵.

Все виды альтерированных ступеней звукоряда представлены на рисунке ниже (рисунок 7).

⁴ Имеется ввиду ассоциация с цветом клавиш фортепиано.

⁵ Если предыдущий знак стоял до этого момента у той же ноты, на той же высоте, в том же такте (понятие такта в настоящей работе не раскрывается подробно за ненадобностью в рамках описания теоретических сведений, связанных с двенадцатитоновой серией).

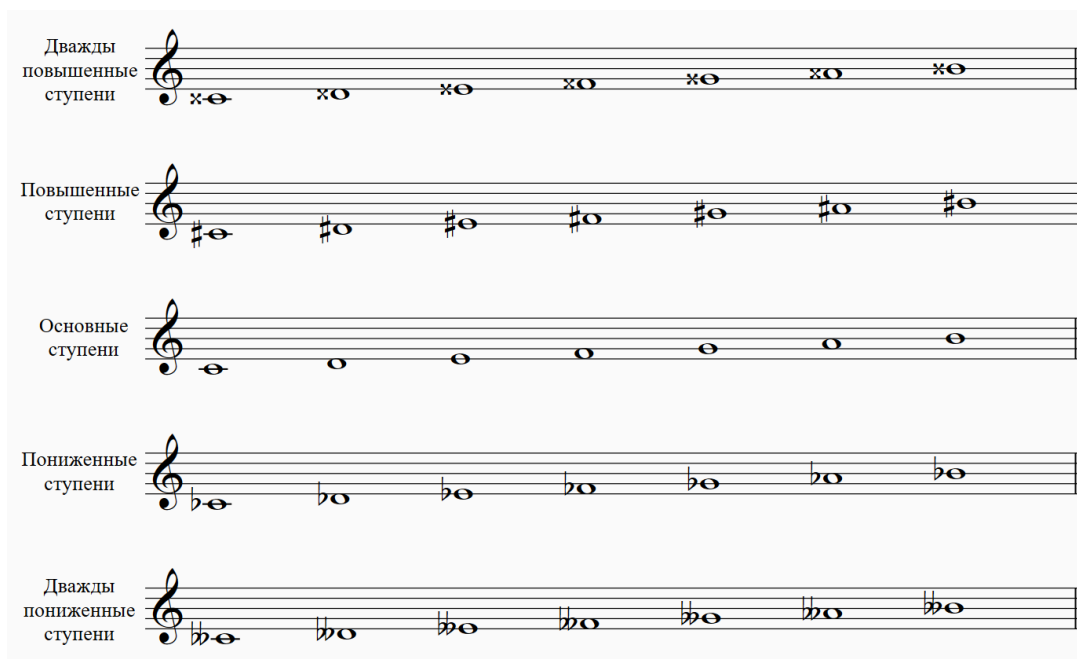


Рисунок 7 – Схема всех видов альтерации всех ступеней звукоряда

Важно заметить, что двойные знаки в самой серии значения иметь не будут – лишь в матрице её форм. Связано это с энгармонизмом – равенством звуков по высоте при различном их значении и обозначении [2].

Например, тон *fisis* будет энгармонически равен *g*: записаны по-разному, но звучание их одинаково.

1.4 Интервалы

Интервалом называется расстояние по высоте между двумя звуками, взятыми последовательно или одновременно. Нижний звук называется его основанием, верхний – его вершиной.

Интервал, звуки которого взяты последовательно (один за другим), называется мелодическим. В зависимости от направления, мелодический интервал может быть восходящим и нисходящим.

Интервалы гармонические интереса для настоящей работы не представляют.

Интервал характеризуется ступеневой величиной, то есть охватываемым им количеством ступеней (рисунок 8).



Рисунок 8 – Схема принципа подсчёта расстояния интервала

Имеется восемь основных названий интервалов, отражающих их величину. Каждое обозначает порядковый номер второго звука интервала так, как если бы от первого его звука брались все ступени до него подряд: прима – 1, секунда – 2, терция – 3, кварта – 4, квинта – 5, секста – 6, септима – 7, октава – 8. Эти интервалы называются простыми. Существуют и составные, превышающие октаву: нона – 9, децима – 10, ундецима – 11, дуодецима – 12, терцдецима – 13, квартдецима – 14, квинтдецима – 15.

Простые интервалы от примы до кварты относят к разряду малых, все прочие (включая составные) – к широким или большим.

Интервалы бывают следующих качеств: а) дважды увеличенные; б) увеличенные; в) большие; г) натуральные; д) малые; е) уменьшённые и ж) дважды уменьшённые.

Ниже приведена таблица интервалов, получающихся между основными ступенями звукоряда (таблица 2).

Таблица 2 – Интервалы между основными ступенями звукоряда

№ п/п	Название интервала	Какими ступенями звукоряда образуется
1	Чистая прима	От данной ступени до её буквального повторения
2	Малая секунда	III – IV, VII – I
3	Большая секунда	I – II, II – III, IV – V, V – VI, VI – VII
4	Малая терция	II – IV, III – V, VI – I, VII – II
5	Большая терция	I – III, IV – VI, V – VII
6	Чистая кварта	I – IV, II – V, III – VI, V – I, VI – II, VII – III

Продолжение таблицы 2

№ п/п	Название интервала	Какими ступенями звукоряда образуется
7	Увеличенная кварта	IV – VII
8	Чистая квинта	I – V, II – VI, III – VII, IV – I, V – II, VI – III
9	Уменьшённая квинта	VII – IV
10	Малая секста	III – I, VI – IV, VII – V
11	Большая секста	I – VI, II – VII, IV – II, V – III
12	Малая септима	II – I, III – II, V – IV, VI – V, VII – VI
13	Большая септима	I – VII, IV – III
14	Чистая октава	От данной ступени до её следующего повторения

Все составные интервалы повторяют свои простые, но вершины каждого из них даются октавой выше.

Качество интервалов изменяется, если в состав их включаются альтерированные ступени. Так, повышение верхнего звука интервала «увеличит» его на «единицу», понижение – «уменьшит». Обратным образом правило распространяется и на нижний звук: при его понижении интервал «увеличится», при повышении – «уменьшится».

К примеру, $c - e$ (большая терция). Понижение верхнего звука даст терцию малую: $c - es$, равно как и повышение нижнего звука: $cis - e$. Двойное понижение верхнего звука даст уменьшённую терцию: $c - eses$ и так далее. Исключение составляют чистые интервалы – они могут быть: дважды увеличенными, увеличенными, чистыми, уменьшёнными и дважды уменьшёнными.

1.5 Октавная система

Существует множество способов визуализировать систему октав, однако наиболее наглядным остаётся обращение к музыкальной клавиатуре, так как её 88 белых и чёрных клавиш соответствуют 88-ми звукам звукоряда и охватывают его почти полностью (рисунок 9).

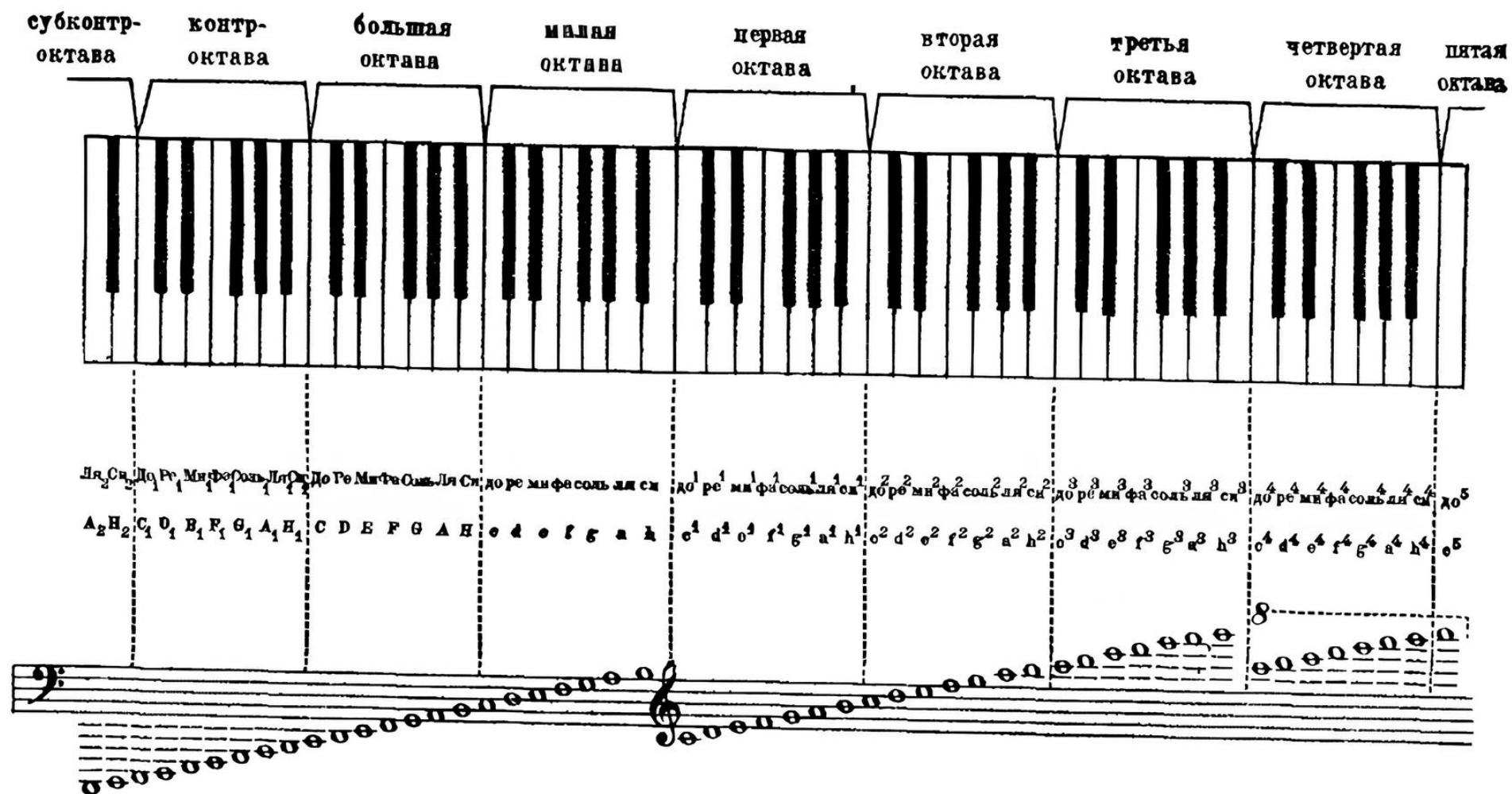


Рисунок 9 – Октавная система

Как видно из рисунка, каждая октава имеет своё название и уникальное обозначение своих ступеней. Система дана по Гельмгольцу [5]. Существует, однако, и другая – научная [6].

В научной системе октавы обозначаются своим порядковым номером, начиная с субконтроктавы (таблица 3).

Таблица 3 – Нотации октавной системы

№ п/п	Наименование октавы	Слоговое обозначение по Гельмгольцу	Буквенное обозначение по Гельмгольцу	Научная нотация
1	Субконтроктава	$До_2 - Си_2$	$C_2 - H_2$	C0 – B0
2	Контроктава	$До_1 - Си_1$	$C_1 - H_1$	C1 – B1
3	Большая октава	$До - Си$	$C - H$	C2 – B2
4	Малая октава	$до - си$	$c - h$	C3 – B3
5	Первая октава	$до^1 - си^1$	$c^1 - h^1$	C4 – B4
6	Вторая октава	$до^2 - си^2$	$c^2 - h^2$	C5 – B5
7	Третья октава	$до^3 - си^3$	$c^3 - h^3$	C6 – B6
8	Четвёртая октава	$до^4 - си^4$	$c^4 - h^4$	C7 – B7
9	Пятая октава	$до^5 - си^5$	$c^5 - h^5$	C8 – B8

Программное обеспечение, ориентированное на оптимизацию процессов, протекающих в музыкальной сфере, использует чаще всего систему научную.

1.6 Серия и матрица её форм

1.6.1 Серия. Формы серии

Настоящая работа касается области одной из композиторских техник музыки XX века. Как известно, эта эпоха явилась временем очередных переломов в истории человечества и повлекла за собой грандиозные по масштабам и качеству перемены. Изменения затронули и искусство, музыкальное – в частности. Научный прогресс открывает дорогу в новые области экспериментов с новыми инструментами, с новыми средствами. Существенным же обстоятельством, радикально изменившим музыкальное искусство, является прорыв в самой технологии сочинения. Это проявляется в постоянном обогащении, в новой организации и использовании компонентов музыкального языка: мелодики, ритмики, гармонии, полифонии, структуры, формы, инструментовки и так далее.

Цтирад Когоутек объясняет это явление так: «Новому содержанию отвечают новые, логически развивающиеся формы музыкального выражения» [7, стр. 26]. Связано это, в том числе, с исчерпанностью традиционной «палитры» средств. На заре новой музыки было изобретено множество принципов композиции, но прошли проверку временем лишь некоторые из них. Например, так называемая серийная техника – это метод сочинения музыки, использующий только высотные серии.

Ц. Когоутек даёт определение серии: «Серия – это ряд звуков, из которого путём его повторения выводится вся высотная ткань произведения» [7, стр. 107]. Иначе говоря, это основа сочинения. Ниже представлен пример такого ряда (рисунок 10).

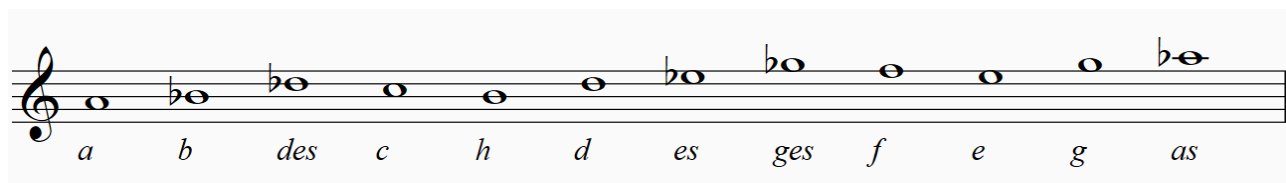


Рисунок 10 – Пример 12-титоновой серии⁶

Так как в этой системе исключаются традиционные художественные методы, особенно распространённые в творчестве композиторов Романтизма и Классицизма, широко применяются средства, видоизменяющие исходный материал – техники нидерландских полифонистов эпохи Возрождения. Антон Веберн, – австрийский композитор и дирижёр, практик в области серийной музыки, – писал: «Итак, ряд есть. Тотчас начинается его видоизменение, развитие. <...> Комбинирующая фантазия нашла следующие формы: противодвижение, обращение, обращение противодвижения. Итого четыре формы» [10, стр. 79].

Так, серия может быть представлена в четырёх формах, а именно: оригинале (Pimus), инверсии (Inversus), ракоходе (Retroversus) и ракоходе-инверсии (Retroinversus) [11].

Ниже дана инверсия примера выше (рисунок 11).

⁶ Данная серия взята из учебного пособия Натальи Сергеевны Гуляницкой (номер 19, страница 205) [11].

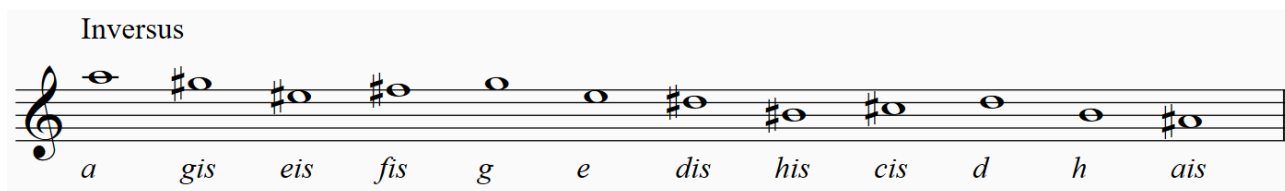


Рисунок 11 – Инверсия

Ниже дан ракоход примера (смотри рисунок 11) (рисунок 12).



Рисунок 12 – Ракоход

Ниже дана ракоход-инверсия примера (смотри рисунок 11) (рисунок 13).

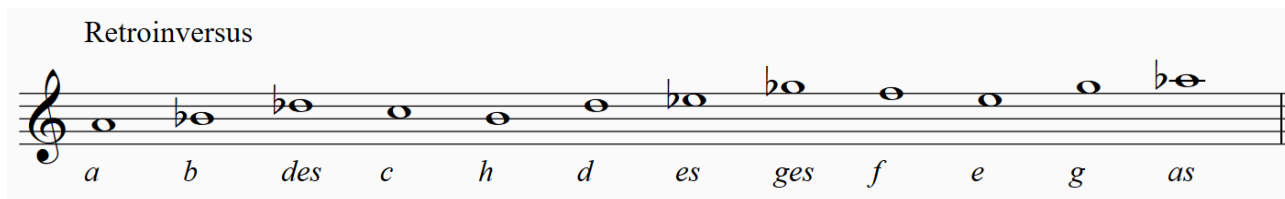


Рисунок 13 – Ракоход-инверсия

Инверсия представляет собой серию, данную в обращении. Это означает, что каждый интервал ряда меняет своё направление на противоположное.

Ракоход – переписанная задом наперёд серия.

Ракоход-инверсия – либо переписанная задом наперёд инверсия, либо инверсия ракохода.

1.6.2 Матрица серийных форм

Матрица серийных форм (называемая ещё «волшебным квадратом») – это матрица, горизонталь которой представляет Р и R, а вертикаль – I и RI⁷. Эта конструкция применяется для удобства анализа серийных композиций и работы с материалом для написания сочинения [11].

⁷ Названия форм даются в сокращении от своих латинских наименований.

Не к каждой серии может быть построена матрица. Существуют критерии оценки качества серии, пригодности её к использованию:

- а) серия не должна быть тождественна хроматической гамме⁸, квартовому или квинтовому кругу⁹;
- б) каждый тон в этом схематическом образовании появляется только однажды;
- в) серия должна иметь определённый план и композиционную направленность, должна подчёркивать некоторые интервалы, характерные для данного ряда, и этим достигать отличия от других серий;
- г) серия не должна слишком выходить за рамки диапазона октавы или ноны;
- д) скачки больше чем на октаву, учитывая возможные октавные транспозиции тонов, не имеют смысла;
- е) не больше двух одинаковых интервалов могут следовать подряд друг за другом;
- ж) малые и большие (широкие) интервалы лучше чередовать;
- з) не допускаются мелодические ходы, создающие арпеджио диатонических аккордов (особенно трезвучий).

Ц. Когоутек замечает: «Вышеописанные правила в музыкальной практике нарушаются, за исключением первых двух, касающихся самой сути додекафонии¹⁰» [7, стр. 124].

Таким образом, нарушение правил а) и б) приведёт к созданию образования, не являющимся серией – соответственно, матрицу построить будет невозможно и даже бессмысленно. Нарушение же остальных, согласно теории, построению матрицы не воспрепятствует.

Пример матрицы форм серии (смотри рисунок 8) дан ниже (рисунок 14).

⁸ Хроматическая гамма – восходящее или нисходящее мелодическое движение по полутонам [8].

⁹ Иначе говоря, идущие друг за другом интервалы чистых кварт и квинт нарушают правило.

¹⁰ Додекафония (от греч. δώδεκα – двенадцать и греч. φωνή – звук) – техника музыкальной композиции, разновидность серийной техники, использующей серии из „двенадцати лишь между собой соотнесённых тонов“» [9].

<i>a</i>	<i>b</i>	<i>des</i>	<i>c</i>	<i>h</i>	<i>d</i>	<i>es</i>	<i>ges</i>	<i>f</i>	<i>e</i>	<i>g</i>	<i>as</i>
<i>gis</i>	<i>a</i>	<i>c</i>	<i>h</i>	<i>ais</i>	<i>cis</i>	<i>d</i>	<i>f</i>	<i>e</i>	<i>dis</i>	<i>fis</i>	<i>g</i>
<i>eis</i>	<i>fis</i>	<i>a</i>	<i>gis</i>	<i>g</i>	<i>ais</i>	<i>h</i>	<i>d</i>	<i>cis</i>	<i>his</i>	<i>dis</i>	<i>e</i>
<i>fis</i>	<i>g</i>	<i>b</i>	<i>a</i>	<i>gis</i>	<i>h</i>	<i>c</i>	<i>es</i>	<i>d</i>	<i>cis</i>	<i>e</i>	<i>f</i>
<i>g</i>	<i>as</i>	<i>ces</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>des</i>	<i>fes</i>	<i>es</i>	<i>d</i>	<i>f</i>	<i>ges</i>
<i>e</i>	<i>f</i>	<i>as</i>	<i>g</i>	<i>fis</i>	<i>a</i>	<i>b</i>	<i>des</i>	<i>c</i>	<i>h</i>	<i>d</i>	<i>es</i>
<i>dis</i>	<i>e</i>	<i>g</i>	<i>fis</i>	<i>eis</i>	<i>gis</i>	<i>a</i>	<i>c</i>	<i>h</i>	<i>ais</i>	<i>cis</i>	<i>d</i>
<i>his</i>	<i>cis</i>	<i>e</i>	<i>dis</i>	<i>d</i>	<i>eis</i>	<i>fis</i>	<i>a</i>	<i>gis</i>	<i>g</i>	<i>ais</i>	<i>h</i>
<i>cis</i>	<i>d</i>	<i>f</i>	<i>e</i>	<i>dis</i>	<i>d</i>	<i>eis</i>	<i>fis</i>	<i>a</i>	<i>gis</i>	<i>h</i>	<i>c</i>
<i>d</i>	<i>es</i>	<i>ges</i>	<i>f</i>	<i>e</i>	<i>g</i>	<i>as</i>	<i>ces</i>	<i>b</i>	<i>a</i>	<i>c</i>	<i>des</i>
<i>h</i>	<i>c</i>	<i>es</i>	<i>d</i>	<i>cis</i>	<i>e</i>	<i>f</i>	<i>as</i>	<i>g</i>	<i>fis</i>	<i>a</i>	<i>b</i>
<i>ais</i>	<i>h</i>	<i>d</i>	<i>cis</i>	<i>his</i>	<i>dis</i>	<i>e</i>	<i>g</i>	<i>fis</i>	<i>eis</i>	<i>gis</i>	<i>a</i>

Рисунок 14 – Пример матрицы серийных форм

Когда речь заходит о работе с додекафоническими произведениями, матрица становится незаменимым и очень удобным инструментом в арсенале будь то композитора, музыковеда или педагога.

2 Постановка задачи и анализ требований

2.1 Постановка задачи

Современный рынок не предоставляет широкого ассортимента программного обеспечения, направленного на решение вопроса о построении матрицы форм серии. Существуют, однако, нотные редакторы, позволяющие производить требуемые операции трансформации двенадцатитонового ряда, но и эти средства имеют ряд недостатков. Помимо прочего они могут не удовлетворять запросам отечественного потребителя, могут быть вовсе недоступны к использованию в связи с какими-либо ограничениями. Например, из-за санкций или нарушений действующего законодательства Российской Федерации. Эти недостатки в разрабатываемом приложении должны быть устранены.

Так, ставится задача создания программного обеспечения, которое:

- а) удовлетворяет функциональным требованиям:
 - 1) принимает данные о серии;
 - 2) позволяет редактировать серию;
 - 3) проводит анализ серии;
 - 4) строит верную матрицу;
 - 5) выводит данные в привычной конечному потребителю системе;
- б) удовлетворяет требованиям к интерфейсу:
 - 1) интуитивно понятный;
 - 2) простой;
- в) удовлетворяет требованиям к производительности;
- г) удовлетворяет требованиям к надёжности;
- д) удовлетворяет требованиям к совместимости;
- е) удовлетворяет требованиям к сопровождению.

Важной составляющей интерфейса должно стать пространство ввода, визуально напоминающее нотный стан. Соответственно, ввод данных должен

производиться либо через загрузку файла, либо с клавиатуры – по усмотрению пользователя.

В качестве удовлетворения пункта е) внутри приложения должна быть встроена информационная панель с инструкцией пользователя и кратким описанием технических возможностей программного обеспечения.

2.2 Анализ систем аналогичного назначения

2.2.1 Анализ программы Sibelius

В отечественной среде профессиональных музыкантов и издателей нотных материалов ещё с начала столетия закрепились программа Sibelius.

Avid Sibelius – редактор нотных партитур для Microsoft Windows, Mac OS, Mac OS X и RISC OS от компании Sibelius Software (ныне часть Avid Technology) [12]. Он широко применяется как в композиторской практике, так и в педагогической, часто является предметом для сравнения с другим подобным программным обеспечением [13, 14].

Предлагает широкий арсенал инструментов для работы с нотным текстом. Допускает ввод тонов на нотный стан посредством клавиатуры или встроенной панели ввода с мгновенным отображением результата. Напоминает текстовые редакторы типа Microsoft Word (рисунок 15).

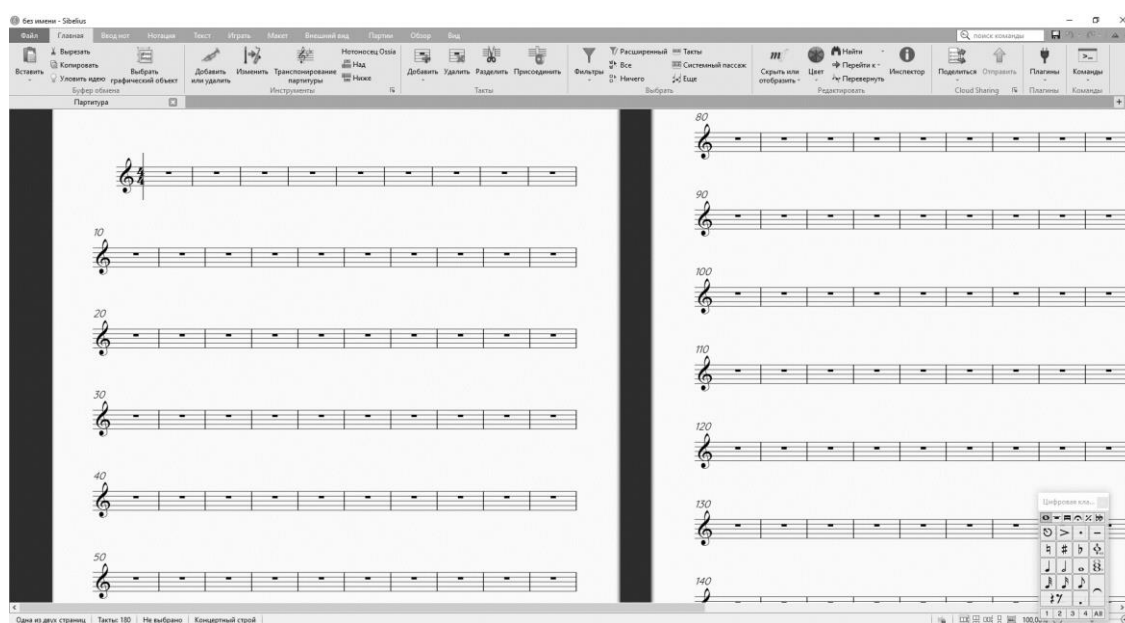


Рисунок 15 – Открывающееся окно программы Avid Sibelius (версии Ultimate)

Матрицу Sibelius построить не сможет, однако имеется возможность произвести преобразование музыкального материала (инверсия, ракоход, ракоход-инверсия).

Официальная покупка и поддержка Sibelius ограничены из-за внешних санкций и внутренней блокировки сайтов компании Avid Technology на территории Российской Федерации.

2.2.2 Анализ программы MuseScore

Muse Score Studio представляет из себя бесплатный WYSIWYG-редактор¹¹ с возможностью быстрого ввода нот как с клавиатуры компьютера, так и с внешней MIDI-клавиатуры [15].

Программа также подвергается техническому анализу как со стороны технических специалистов, так и со стороны рабочих в сфере музыкального исполнительства, композиции и педагогики [16].

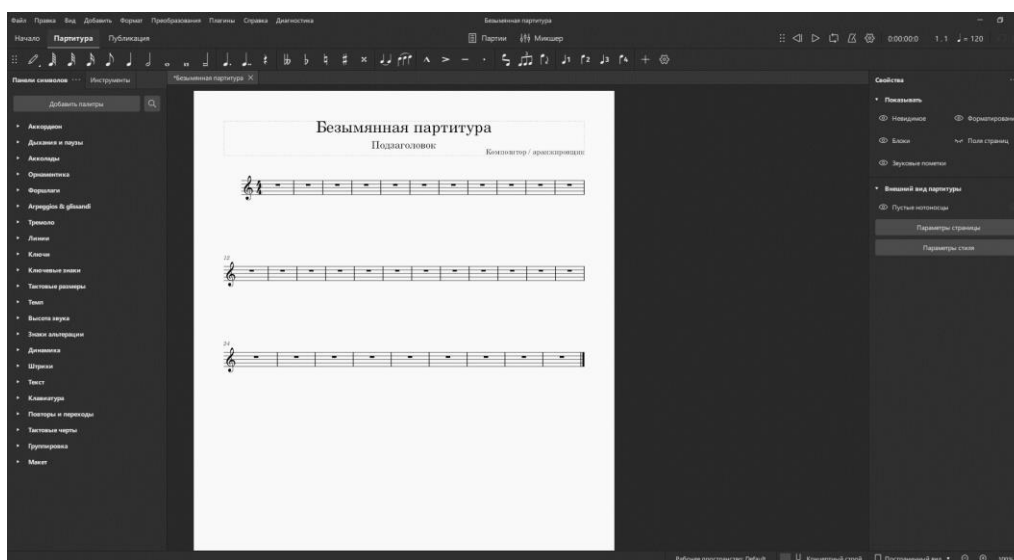


Рисунок 16 – Открывающееся окно программы MuseScore Studio 4

Возможности, что и у Avid Sibelius – те же. Разве что последний гораздо больше ценится в профессиональной среде композиторов, аранжировщиков и издателей.

¹¹ WYSIWYG (от англ. «What You See Is What You Get» – «что видишь, то и получишь») – свойство прикладных программ или веб-интерфейсов, в которых содержание отображается в процессе редактирования и выглядит максимально близко похожим на конечную продукцию, которая может быть печатным документом, веб-страницей или презентацией [17].

2.2.3 Анализ приложения ToneMatrix

ToneMatrix – любительское оконное приложение за авторством fonazza-stent, пользователя веб-платформы для размещения, совместной разработки и скачивания программного обеспечения с открытым программным кодом Source Forge [18]. Приложение предоставляет возможность генерации случайной серии и её форм, а также генерации матрицы форм введенной пользователем серии (рисунок 17).

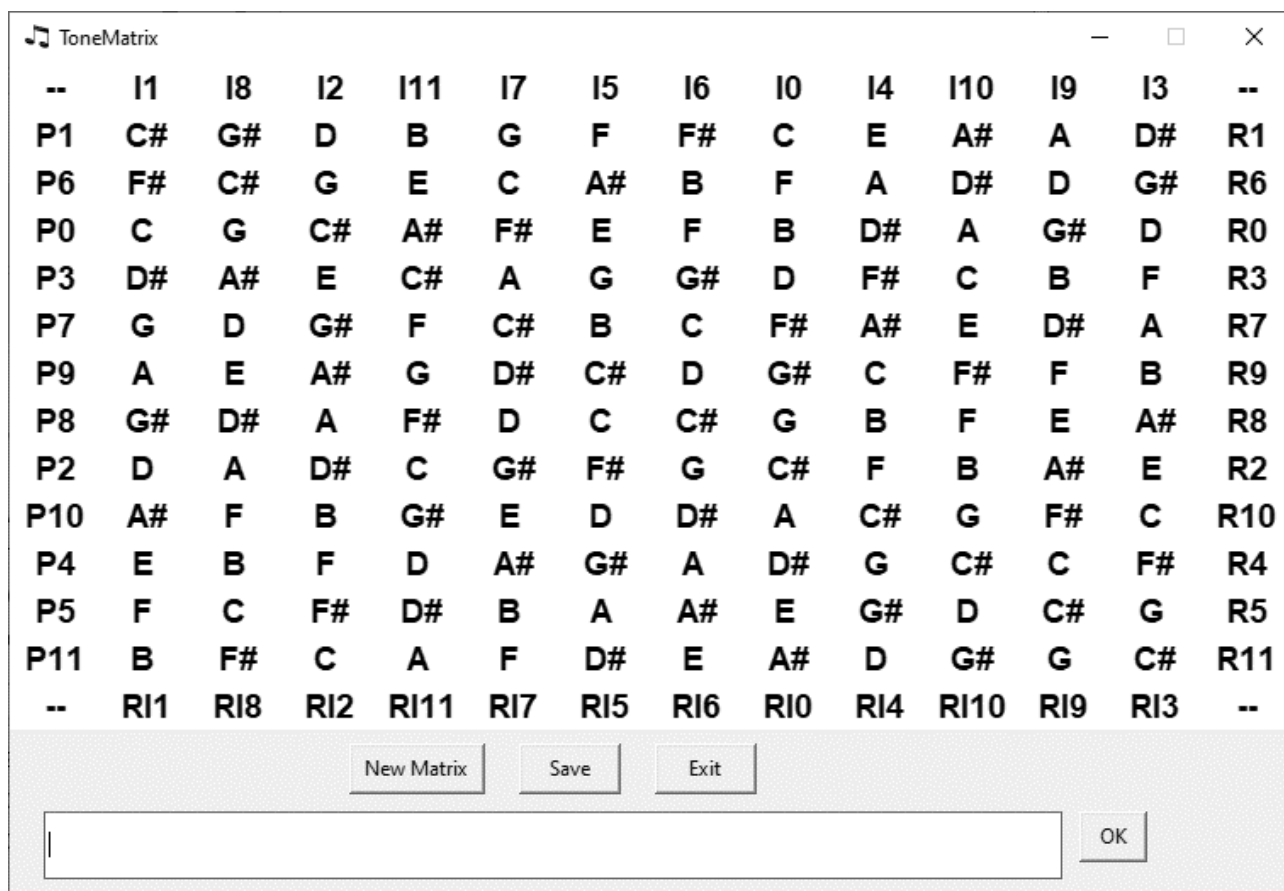


Рисунок 17 – Приложение ToneMatrix

Названия тонов выводятся в американской системе. Программа не поддерживает никаких знаков, кроме диеза и бекара.

2.2.4 Сравнительный анализ аналогичных систем

В результате изучения возможностей, предлагающихся рассмотренными выше примерами программного обеспечения аналогичного разрабатываемому

приложению назначения, было проведено качественное сравнение данных приложений.

Sibelius – профессиональный редактор партитур с широким арсеналом средств для набора нот и других графических представлений музыкального материала любой сложности. Специального функционала для построения матрицы форм серии не имеет, однако располагает инструментарием для трансформации двенадцатитонового ряда. К сожалению, сайты компании Avid заблокированы на территории Российской Федерации, а потому приобрести лицензию оказывается затруднительно.

MuseScore распространяется бесплатно и позиционируется как «молодая» замена Sibelius. Функционал у приложения не многим уступает предшественнику, а интерфейс интуитивно понятен. Специального функционала для построения матрицы серийных форм нет. Есть, однако, возможность бесплатной установки расширения для трансформации музыкального материала, что может позволить построить основные формы серии (равно как и Sibelius).

ToneMatrix является любительским приложением с довольно ограниченным функционалом. Оно позволяет построить матрицу и даже сгенерировать случайную, но: а) делает это строго в американской системе; б) не работает ни с какими знаками альтерации, кроме как с бекарами и диезами; в) сохраняет матрицу только в виде изображения.

Качественное сравнение аналогов представлено ниже (таблица 4).

Таблица 4 – Сравнительная таблица аналогов разрабатываемой системы

№ п/п	Критерий	Avid Sibelius	Muse Score Studio	ToneMatrix	Разрабатываемая система
1	Есть специальная функция построения матрицы	-	-	+	+
2	Отображение названий тонов в немецкой системе	-	-	-	+
3	Бесплатный доступ к функционалу	-	+	+	+

Продолжение таблицы 4

№ п/п	Критерий	Avid Sibelius	Muse Score Studio	ToneMatrix	Разрабатываемая система
4	Есть представление нотного стана для ввода тонов	+	+	-	+
5	Поддержка сохранения матрицы в текстовом файле	-	-	-	+
6	Строгий учёт интервалики при построении матрицы	-	-	-	+
7	Поддержка всех знаков альтерации	+	+	-	+
8	Поддержка всех форм преобразования исходной серии	+	+	+	+
9	Поддержка функции анализа серии	-	-	-	+
10	Отечественный продукт	-	-	-	+
11	Поддерживается на территории Российской Федерации	-	+	+	+

Примечания к таблице: «+» – означает наличие функции, «-» – отсутствие.

Таблица показывает, что требуемый функционал для решения поставленной задачи выполняется рассматриваемыми аналогами либо частично, либо не выполняется вовсе. Все эти ограничения разрабатываемое приложение должно ликвидировать.

2.3 Формулировка требований к разрабатываемому приложению

В результате проведённых исследований аналогов и выявления их недостатков были сформулированы требования к разрабатываемому приложению. Приложение должно:

- а) принимать данные о серии:
 - 1) через файл;
 - 2) вводом с клавиатуры;

- б) предоставлять возможность редактировать серию;
- в) проводить достаточный анализ серии;
- г) строить верную матрицу форм серии;
- д) выводить данные о тонах в немецкой системе нотации.

Для наглядного отображения функционала разрабатываемого приложения было принято решение использовать диаграмму вариантов использования из унифицированного языка моделирования.

Унифицированный язык моделирования (Unified Modeling Language, UML) является графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению. С помощью UML можно разработать детальный план создаваемой системы, содержащий не только её концептуальные элементы, такие как системные функции и бизнес-процессы, но и конкретные особенности, например классы, написанные на специальных языках программирования, схемы баз данных и программные компоненты многократного использования.

Диаграммы вариантов использования – основной вид диаграмм при моделировании поведения системы, подсистемы или класса. Каждая из них показывает набор вариантов использования и действующих лиц в их взаимодействии.

Диаграммы вариантов использования важны для визуализации, специфицирования и документирования поведения элемента. Они обеспечивают доступность и понятность систем, подсистем и классов за счет внешнего представления того, как эти элементы могут быть использованы в контексте. Кроме того, такие диаграммы важны для тестирования работающих систем посредством прямого проектирования и для обеспечения их понимания посредством обратного проектирования [19].

Исходя из описанных выше определённых ранее функциональных требований к разрабатываемому приложению была создана UML-диаграмма вариантов использования (рисунок 18).

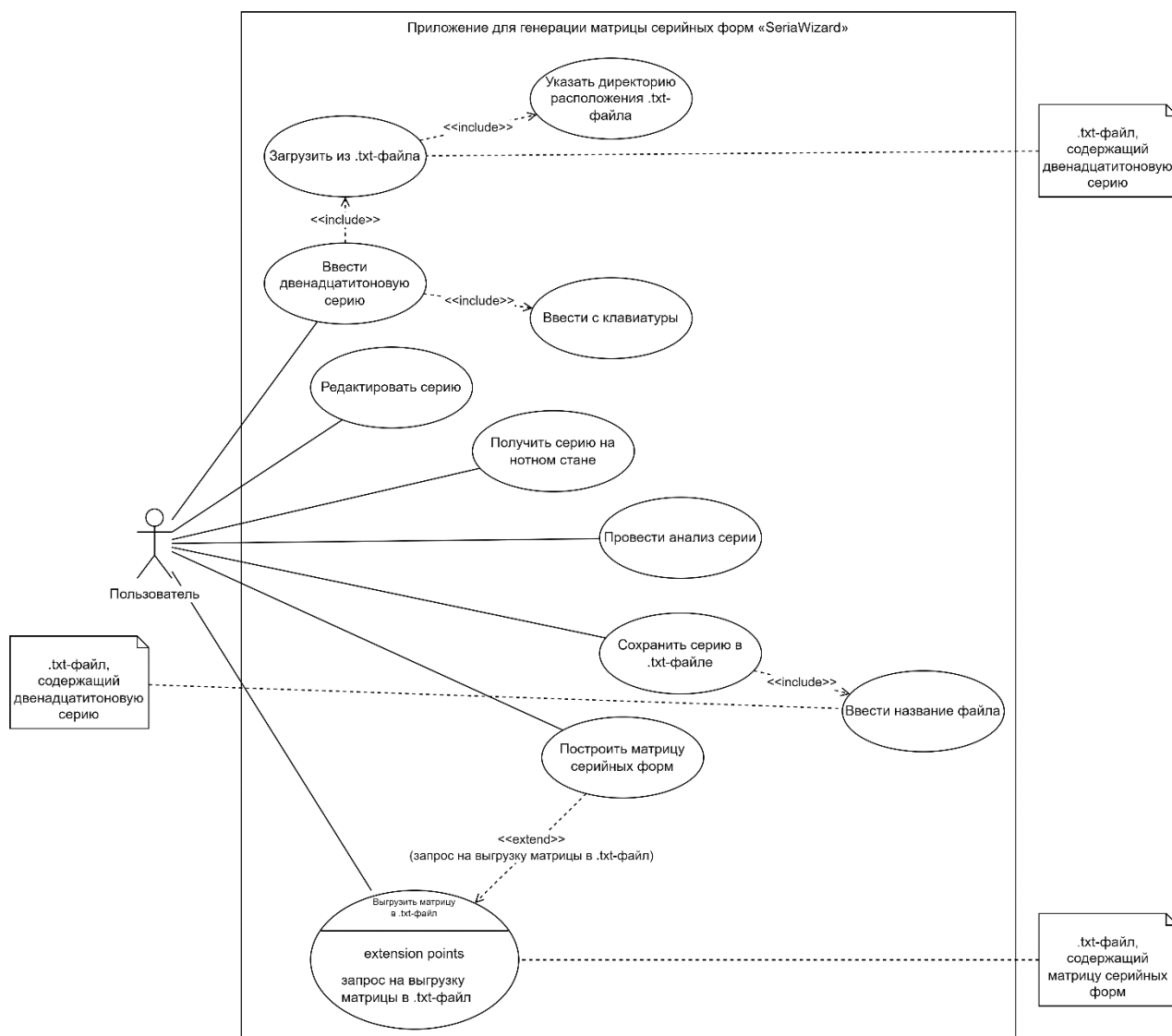


Рисунок 18 – UML-диаграмма вариантов использования приложения для генерации матрицы серийных форм «SeriaWizard»

Для большей наглядности, а также для более чёткого понимания функциональных требований разрабатываемого приложения, были описаны сценарии основных вариантов использования, среди которых:

- введение двенадцатитоновой серии;
- проведение анализа серии;
- построение матрицы серийных форм.

Сценарий для варианта использования «Ввести двенадцатитоновую серию» дан на рисунке ниже (рисунок 19).

Прецедент: Ввести двенадцатитоновую серию	
ID: 1	
Краткое описание: Пользователь вводит новую двенадцатитоновую серию	
Главные актёры: Пользователь	
Второстепенные актёры: НЕТ	
Предусловие: 1 Пользователь открыл приложение	
Основной поток: 1 Если пользователь выбирает создать новую серию: 1.1 Серия принимается с клавиатуры 2 Иначе, если пользователь выбирает загрузить из файла: 2.1 Принимается директория текстового документа формата .txt 2.2 Если файл и данные в нём успешно читаются, то: 2.2.1 Приложение загружает серию в редактор 2.3 Иначе: 2.2.1 Приложение выводит ошибку о провале операции	
Постусловия: Загруженная серия временно сохраняется внутри программы	
Альтернативные потоки: НЕТ	

Рисунок 19 – Сценарий варианта использования приложения для генерации матрицы серийных форм «SeriaWizard» «Ввести двенадцатитоновую серию»

Вариант использования «Ввести двенадцатитоновую серию» описывает процесс принятия в систему данных о серии через ввод с клавиатуры и текстовый файл.

Сценарий для варианта использования «Провести анализ серии» дан на рисунке ниже (рисунок 20).

Прецедент: Провести анализ серии	
ID: 7	
Краткое описание: Пользователь запускает вычислительный процесс анализа серии	
Главные актёры: Пользователь	
Второстепенные актёры: НЕТ	
Предусловие: 1 Пользователь загрузил серию	
Основной поток: 1 Пользователь запускает алгоритм анализа серии 2 Приложение выводит результат анализа	
Постусловия: Упрощённый результат анализа временно сохраняется в системе	
Альтернативные потоки: НЕТ	

Рисунок 20 – Сценарий варианта использования приложения для генерации матрицы серийных форм «SeriaWizard» «Провести анализ серии»

Вариант использования «Провести анализ серии» описывает процесс анализа серии, проверку каждого из правил её качества.

Сценарий для варианта использования «Построить матрицу серийных форм» дан на рисунке ниже (рисунок 21).

Прецедент: Построить матрицу серийных форм	
ID: 8	
Краткое описание: Пользователь запускает процесс построения матрицы серийных форм	
Главные актёры: Пользователь	
Второстепенные актёры: Нет	
Предусловие:	
1	Пользователь загрузил серию
2	Результат анализа достаточно успешный для построения матрицы
Основной поток:	
1	Пользователь запускает алгоритм построения матрицы
2	Приложение выводит матрицу на экран
Постусловия: Матрица временно хранится в системе	
Альтернативные потоки: Нет	

Рисунок 21 – Сценарий варианта использования приложения для генерации матрицы серийных форм «SeriaWizard» «Построить матрицу серийных форм»

Вариант использования «Построить матрицу серийных форм» описывает процесс построения серии.

В ходе анализа функциональных требований к разрабатываемому приложению была создана диаграмма вариантов использования, а также выделены основные её элементы и их спецификации, которые отражают ход взаимодействия пользователя с приложением.

Нефункциональные требования описывают такие характеристики системы, которые не связаны с её функционалом. В ходе разработки персональной системы напоминаний пользователя были определены следующие нефункциональные требования:

- удобство и простота использования – интерфейс системы должен быть интуитивно понятен и прост в использовании;

- доступность – система должна быть доступна с минимальным временем простоя;
- производительность – система должна обеспечивать быстрое время отклика;
- расширяемость – архитектура системы должна быть гибкой, чтобы позволять добавление новых функций.

Таким образом были сформулированы требования к разрабатываемому приложению.

2.4 Выбор и обоснование технологии

Приложение должно работать в операционной системе Windows, так как именно этот продукт широко охватывает электронно-вычислительные машины, представленные в консерваториях, музыкальных училищах и многих других местах, где предполагаемый потребитель будет пользоваться разрабатываемым приложением [21].

Исходя из вышеопределённых требований, можно сказать, что достаточным будет приложение консольное, с налаженной графической частью. Средства Windows предлагают широкий спектр инструментов для реализации задачи подобного рода [22].

В качестве среды разработки будет использоваться Microsoft Visual Studio как наиболее подходящая и удобная для написания приложений под операционную систему Windows.

Язык программирования – C++, так как это язык низкого уровня, а значит с ним открывается доступ ко всем ресурсам электронно-вычислительной машины. Он активно применяется во всех отраслях науки, где требуется автоматизация вычислительного процесса [23, 24].

3 Проектирование приложения

3.1 Проектирование структуры данных

Для точного описания архитектуры данных было принято решение пользоваться терминологией UML. Для фиксирования словаря разрабатываемой системы целесообразно пользоваться понятием класс.

Грэди Буч даёт следующее определение понятия класс: «Класс – это описание множества объектов с одинаковыми атрибутами, операциями, связями и семантикой. Хорошо структурированные классы имеют четко очерченные границы и формируют сбалансированное распределение обязанностей в системе» [19, стр. 62].

Центральными понятиями из аппарата музыкальной теории для описания серии, её форм и матрицы её форм являются тон (нота) и интервал. Именно эти объекты и станут главными классами архитектуры для реализации алгоритмов анализа и построения матрицы форм серии.

Стоит также выделить методы для специализированной обработки данных о тонах в отдельный класс.

Переменные, которые потребуется использовать несколькими типами операций, разумно тоже описать в отдельном классе.

Потребуется выводить на экран достаточно ясный, эстетически приемлемый и достаточный интерфейс – для этой задачи тоже потребуются отдельные классы. Равно как и для подготовки строк к выводу в консоли Windows (требуется настройка верного чтения кириллицы).

Были выделены и иные методы, требующие выделения в отдельный класс архитектуры:

- а) для работы с файловой системой;
- б) для подготовки данных к сохранению в файловой системе;
- в) для подготовки данных к загрузке в систему.

Для более наглядного представления архитектуры классов разрабатываемого приложения была создана UML-диаграмма классов (рисунок 22).

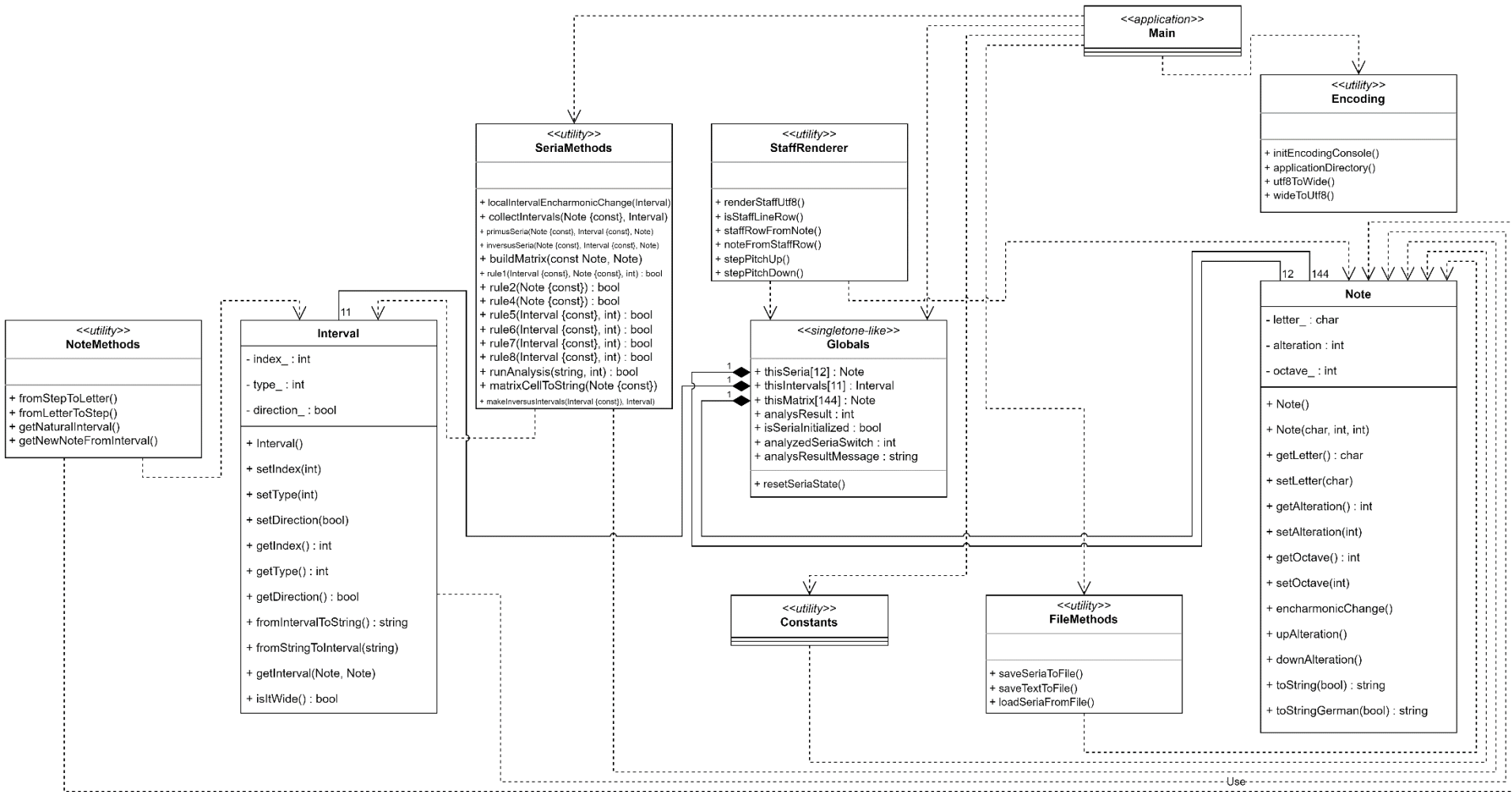


Рисунок 22 – UML-диаграмма классов системы приложения для генерации матрицы серийных форм «SeriaWizard»

Класс «Note» содержит все необходимые для решения задачи характеристики тона: символ («letter») (в диапазоне от А до G, согласно американской системе нотации), знак альтерации («alteration») и значение октавы («octave»). В целях снятия лишней нагрузки на систему параметр «letter» было принято сделать символьного типа (char). «alteration» же лучше всего сделать целочисленного типа – для упрощения дальнейшей алгоритмизации вычислительных операций с этим параметром. Можно предположить, что будет представлять каждое из значений «alteration»:

- дубль-диез – 2;
- диез – 1;
- бекар – 0;
- бемоль – минус 1;
- дубль-бемоль – минус 2.

Помимо традиционных методов (например, так называемых «геттеров» и «сеттеров») потребуются и специальные:

- а) для совершения энгармонической замены тона («enharmonicChange»);
- б) для повышения и понижения тона («upAlteration» и «downAlteration»);
- в) для возврата записи названия тона в американской и немецкой системах нотации в виде строки («toString» и «toStringGerman»).

Вторым основополагающим классом является «Interval». Он аналогичным образом содержит необходимые для описания интервала параметры: индекс («index»), качество («type») и направление («direction»). Значение первого из них ассоциируется со шкалой Танеева, как более пригодной к вычислениям, чем общепринятая¹². Качество так же выражено шкалой:

- увеличенный – 2;
- большой – 1;

¹² Каждый интервал обозначается цифрой, выражающей количество заключающихся в нём единиц. Прима обозначается нулём, так как в ней это количество равняется нулю [20].

- чистый – 0;
- малый – минус 1;
- уменьшённый – минус 2.

Дважды увеличенными и дважды уменьшёнными интервалами можно пренебречь, так как в исходной серии таковых быть не может в силу бессмысленности такого предположения. Связано это опять же с энгармонизмом.

Теперь к вопросу о направлении. Параметр «direction» должен быть логического типа, так как направлений у интервала два: восходящее и нисходящее. Исключение составляет чистая прима – она как такового направления не имеет, потому что представляет собой точное повторение тона. Исходя из правила б) (смотри главу 1.6.2), это исключение программной обработки в классе «Interval» не требует.

Последним классом, требующим особого внимания, является «SerialMethods». Он содержит методы для проведения анализа по каждому из правил серии, кроме в) – рассмотрение его в автоматизации не нуждается, так как центральная задача разрабатываемого приложения не проведение художественного анализа качества серии, а анализ её достаточной пригодности к применению на практике.

Стоит сказать также об избыточности (в рамках поставленной задачи) использования функций для специального построения ракохода и ракохода-инверсии. Для создания матрицы достаточно методов:

- а) строящего основной вид серии от конкретного тона (метод «primusSerial»);
- б) строящего инверсию от конкретного тона (метод «inversusSerial»).

Все прочие классы предназначены для обеспечения корректной работы с файловой системой электронно-вычислительной машины и корректного вывода данных на экран с учётом базовых эстетических потребностей будущего пользователя разрабатываемого приложения, а также не нарушающих порядок работы с приложением.

3.2 Проектирование алгоритмов

Для наглядной иллюстрации принципа работы алгоритмов разрабатываемого приложения было принято решение снова обратиться к UML, а именно – к диаграммам деятельности и последовательности.

Г. Буч даёт следующее определение диаграмме деятельности: «Диаграммы деятельности – это один из пяти видов диаграмм, применяемых в UML для моделирования динамических аспектов систем. По сути, диаграмма деятельности представляет собой блок-схему, которая показывает, как поток управления переходит от одной деятельности к другой» [19, стр. 281].

Он же описывает и диаграмму последовательности: «Диаграмма последовательности – это диаграмма взаимодействия, которая подчеркивает временной порядок сообщений. Изображается как таблица, в которой представлены объекты, расположенные вдоль оси X, и сообщения, упорядоченные по ходу времени – вдоль оси Y» [19, стр. 264].

Первый алгоритм, требующий комментария, описан в классе «SerialMethods» – «primusSeria» (смотри рисунок 22). Именно он заполняет входящую матрицу Primus от также поданного на вход тона. Была составлена диаграмма деятельности метода (рисунок 23).

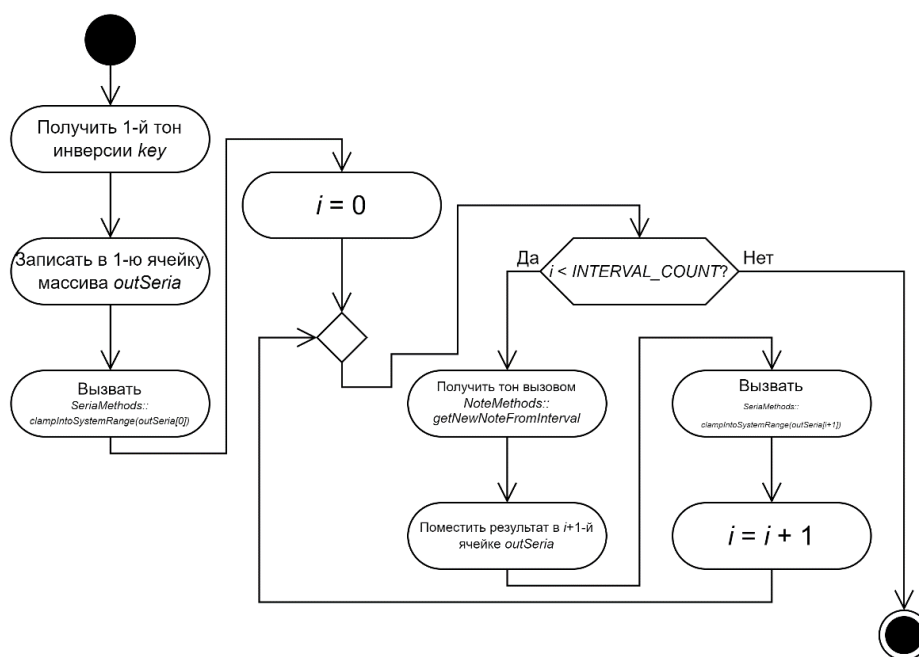


Рисунок 23 – UML-диаграмма деятельности метода «primusSeria»

Для более наглядного представления взаимодействия между классами была создана диаграмма последовательности (рисунок 24).

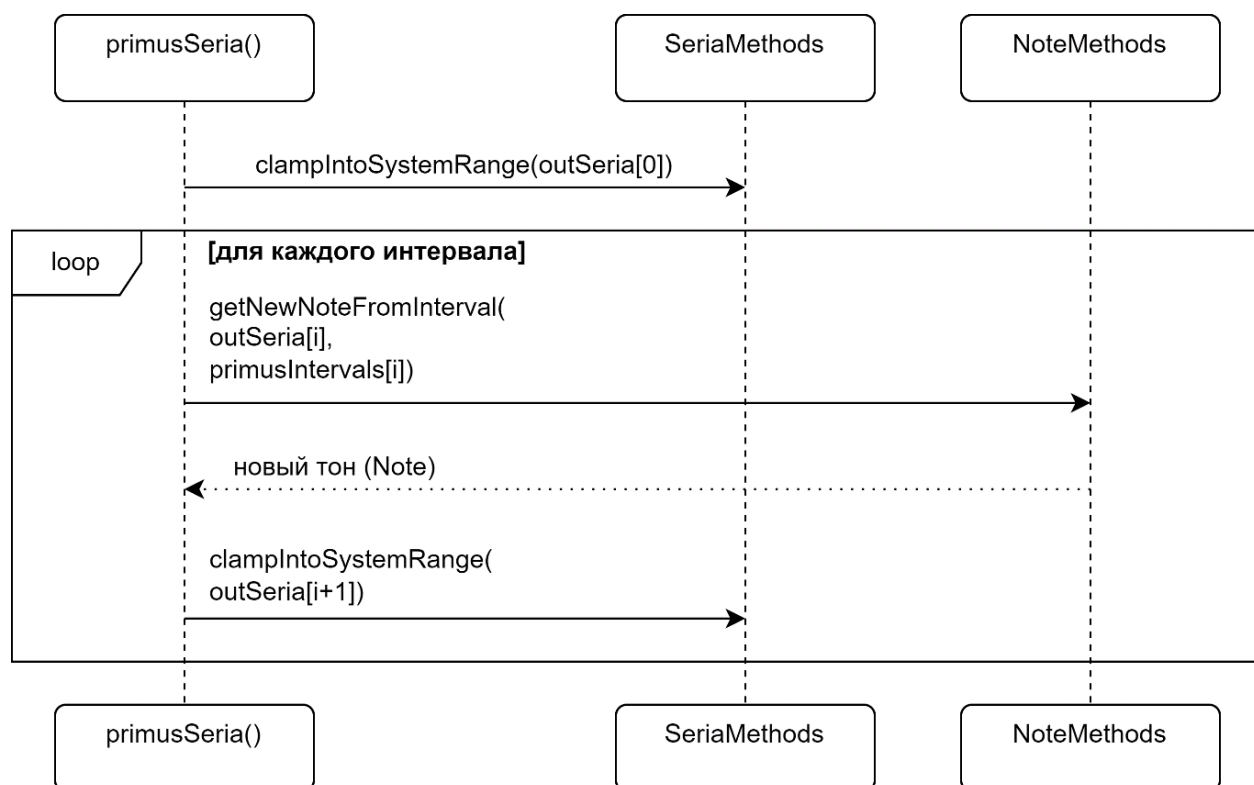


Рисунок 24 – UML-диаграмма последовательности метода «primusSeria»

Как можно заметить, метод обращается к методу как своего класса, так и к методу класса «NoteMethods». Функционал последнего позволяет получить ноту через интервал от данной. Без этой операции работа «primusSeria» не представляется возможной.

Следующий важный для построения матрицы серийных форм метод – «inversusSeria». Он работает аналогичным образом, но заполняет подаваемую на вход матрицу инверсией от данного звука. Сама инверсия производится специальным методом «makeInversusIntervals», который, в свою очередь, обращается к методу класса «Interval». Происходит инверсия каждого интервала в последовательности, а затем через «primusSeria» строится новая форма исходного ряда.

Для более наглядного представления функционала метода была создана диаграмма последовательности (рисунок 25).

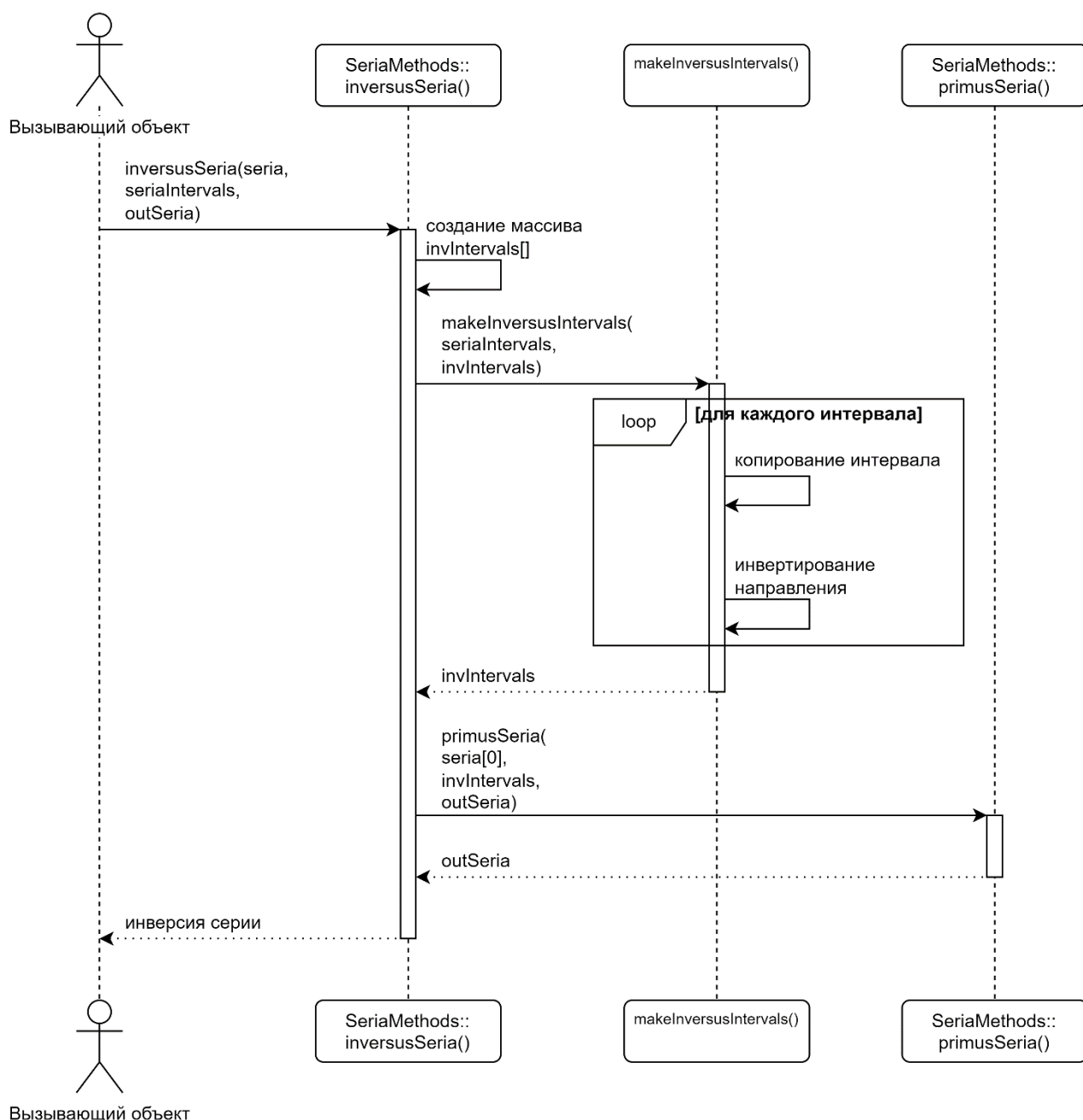


Рисунок 25 – UML-диаграмма последовательности метода «inversusSeria»

Этих методов вполне достаточно для построения матрицы форм серии, поэтому нет необходимости реализовывать алгоритмы вычисления ракохода и ракохода-инверсии.

Следующим алгоритмом, требующим пояснения является «buildMatrix», собственно строящий матрицу. Диаграмма деятельности его представлена на рисунке ниже (рисунок 26).

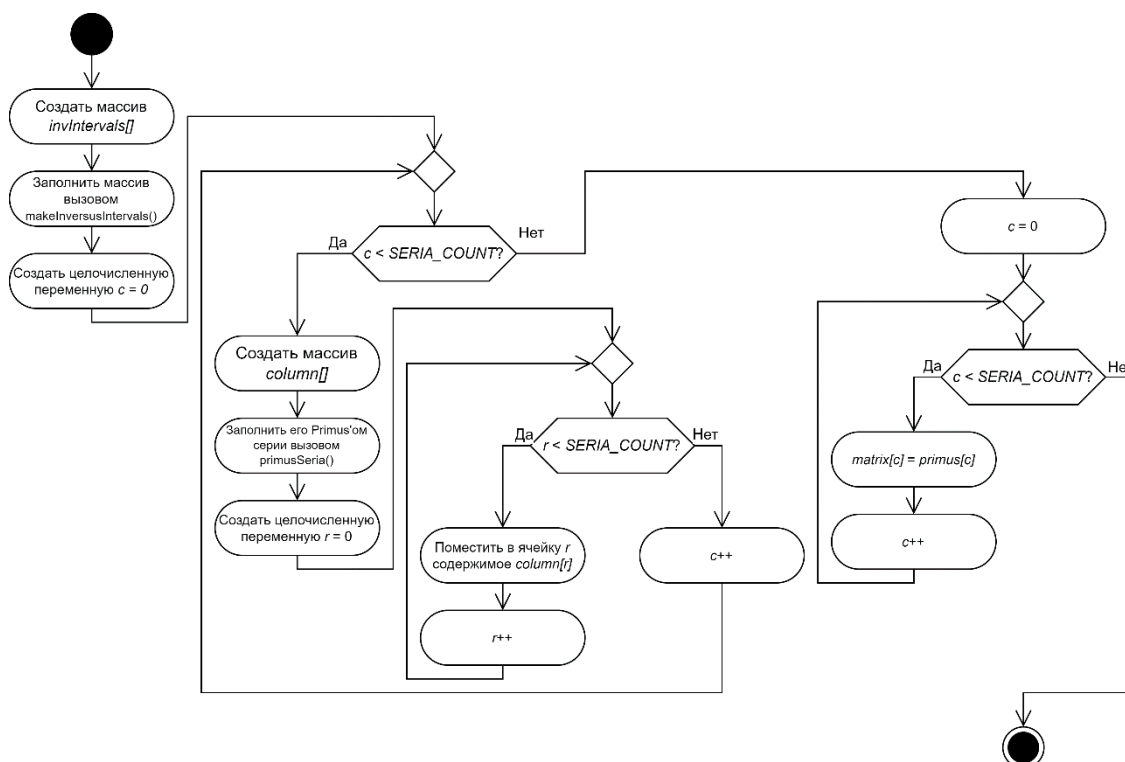


Рисунок 26 – UML-диаграмма деятельности метода «buildMatrix»

Так как метод часто обращается к другим, имеет смысл проиллюстрировать его работу также в виде диаграммы последовательности (рисунок 27).

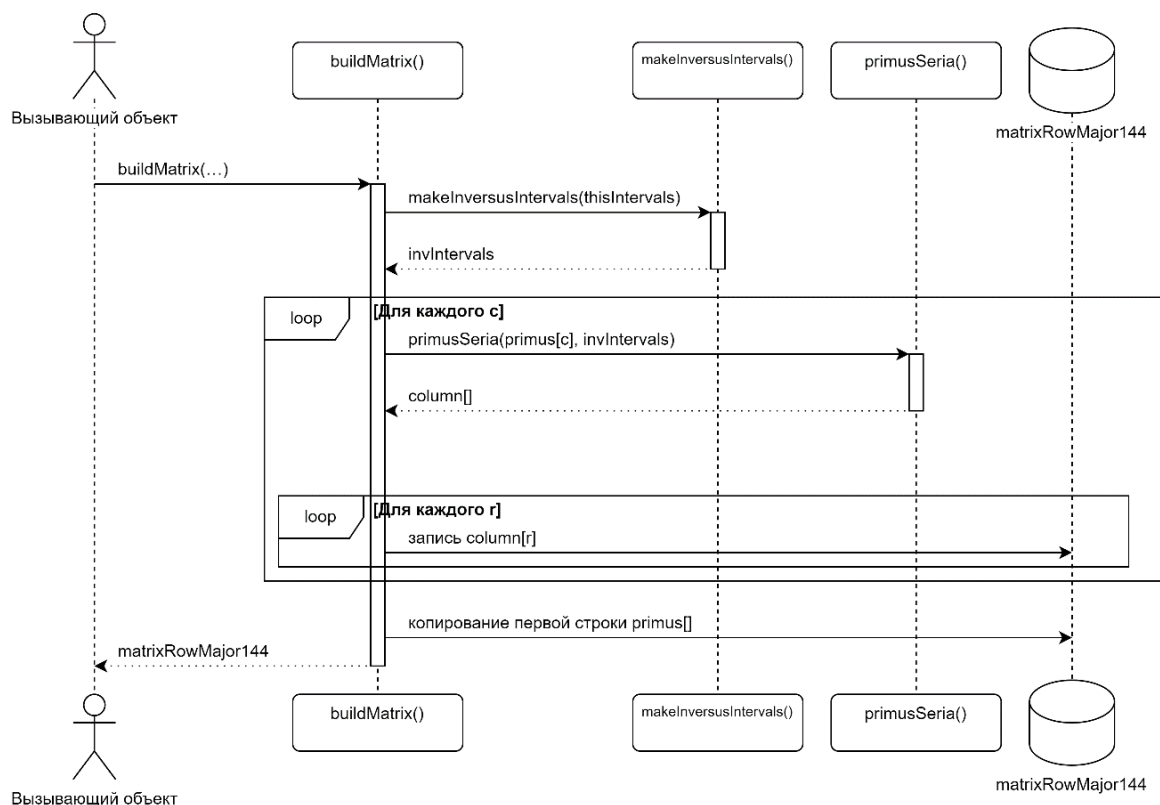


Рисунок 27 – UML-диаграмма последовательности метода «buildMatrix»

Сперва формируется набор инвертированных интервалов на базе массива, образованного от `Primus` и хранящегося в системе в качестве глобальной переменной. Так получается первый столбец матрицы. Затем каждая строка заполняется методом «`primusSeria`», в который в качестве начального тона передаётся содержимое первой ячейки.

Так получается корректная матрица серийных форм, передаваемая в конце работы метода в массив «`matrixRowMajor144`».

3.3 Проектирование интерфейса

Следующей задачей является проектирование интерфейса. Этот этап представляется одним из важнейших при разработке программного продукта. Плохое обеспечение графических элементов управления приложения может по меньшей мере оттолкнуть пользователя, а в худшем случае и вовсе начать мешать работе.

В практике разработки визуальных сред широко применяются методы UX- и UI-дизайна¹³. В рамках настоящей работы было принято решение обратиться к методу вайрфреймов.

Вайрфреймы¹⁴ (англ. Wireframes) – это схематичные, низко- или среднедетализированные визуальные представления интерфейса, которые фокусируются на структуре, информационной иерархии, расположении элементов, функциональности и основных взаимодействиях, не акцентируя внимание на визуальном стиле, цветах или графике [25]. Этого метода будет вполне достаточно для иллюстрации взаимодействия между окнами разрабатываемого приложения.

Было создано представление дизайна пользовательского интерфейса (рисунок 28).

¹³ UX/UI – это проектирование пользовательского опыта (User Experience) и пользовательского интерфейса (User Interface), которое делает цифровой продукт удобным, понятным и визуально приятным. UX отвечает за логику взаимодействия, сценарии и пользовательский опыт, а UI – за визуальный интерфейс: кнопки, цвета, типографику и элементы взаимодействия. UX/UI-дизайн используют при создании сайтов, мобильных приложений, веб-сервисов и цифровых платформ [26].

¹⁴ Далее он будет называться в переводе на русский язык – представлением дизайна пользовательского интерфейса.

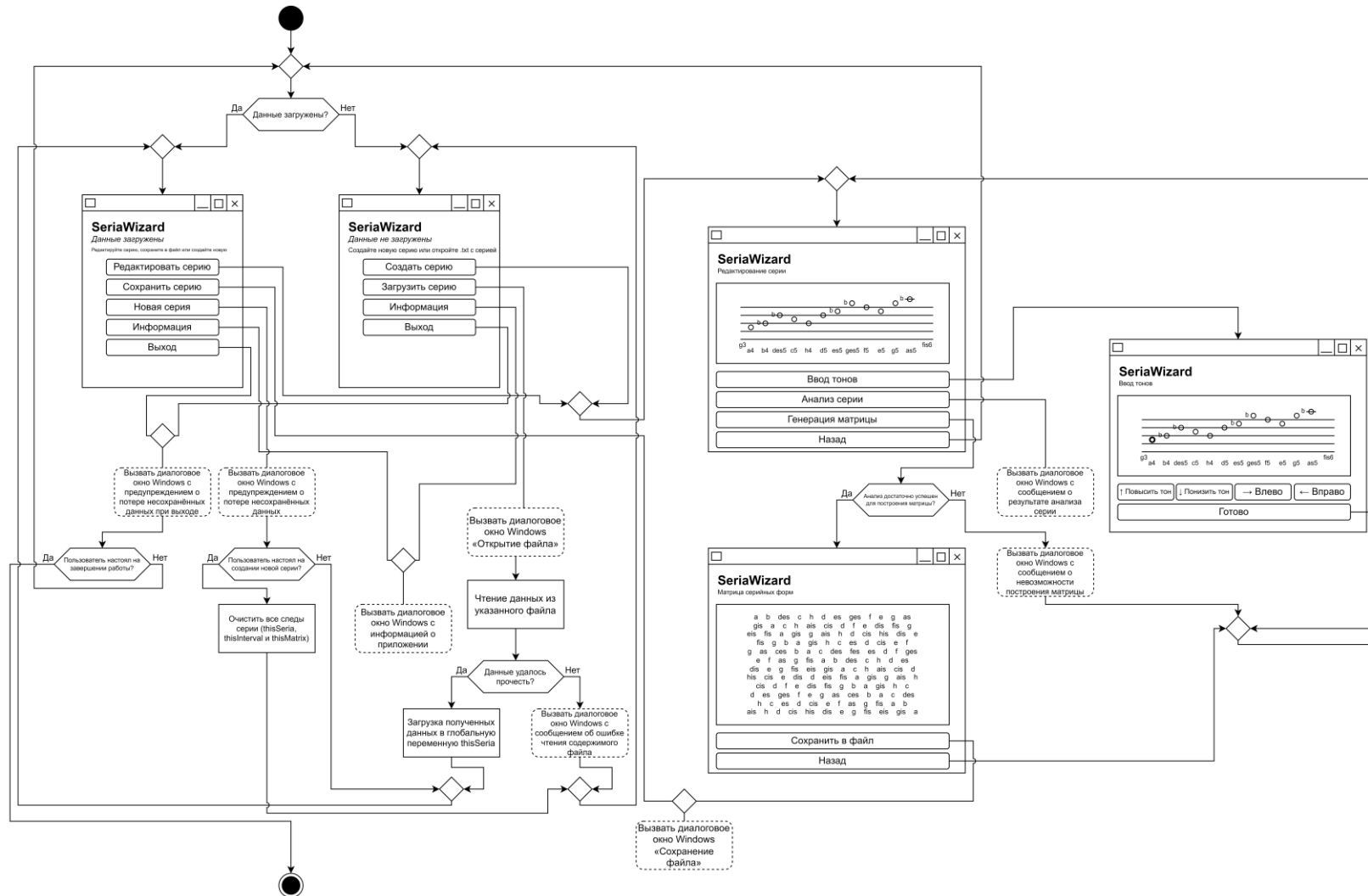


Рисунок 28 – Представление дизайна пользовательского интерфейса приложения для генерации матрицы серийных форм «SeriaWizard»

В рамках, отрисованных пунктиром, описаны действия, связанные с обращением к средствам операционной системы Windows. Некоторые связи, ведущие к ним, не имеют на концах стрелочек: в данном случае имеется ввиду возможность возврата к предыдущему окну приложения при выходе из диалогового Windows-окна.

Итак, пользователя встречает так называемое главное окно, своего рода основное меню приложения (рисунок 29). Оно представлено в двух вариантах: первый (слева на рисунке) – в случае, если данные уже загружены; второй (справа на рисунке) – если данных ещё в системе нет.

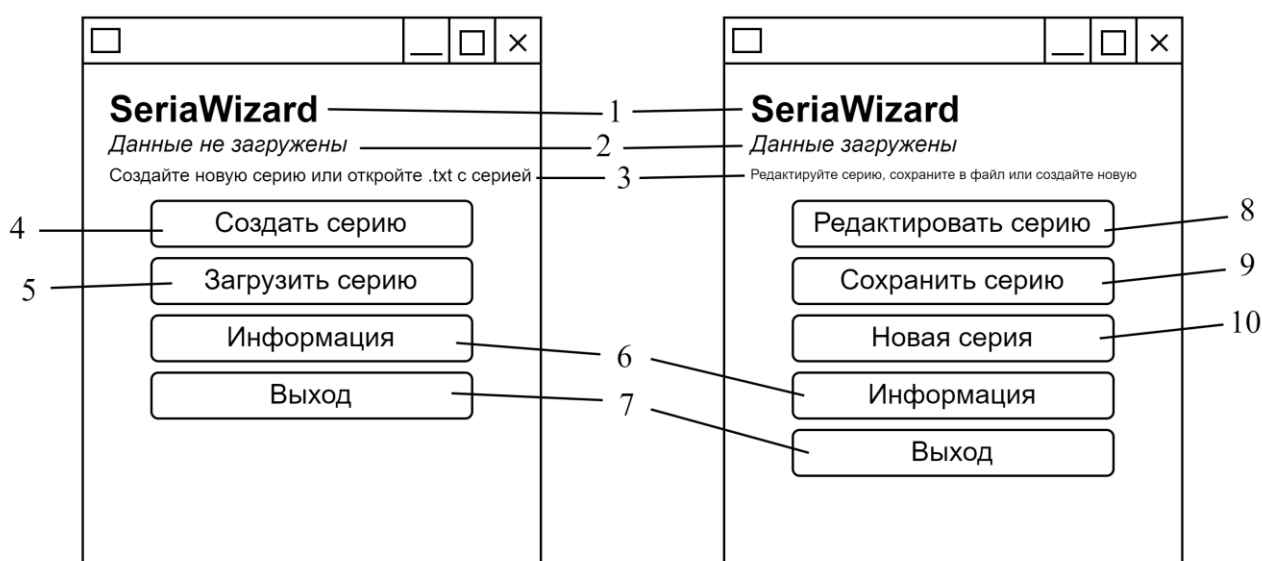


Рисунок 29 – Схема главного окна приложения

Основные элементы данного окна:

- 1 – надпись с названием приложения;
- 2 – строка состояния наличия данных;
- 3 – надпись с краткой инструкцией;
- 4 – кнопка для создания новой серии;
- 5 – кнопка для загрузки серии из файла;
- 6 – кнопка для вывода информации о приложении;
- 7 – кнопка для завершения работы приложения;
- 8 – кнопка для перехода к редактированию серии;
- 9 – кнопка для сохранения серии в файл;

10 – кнопка для создания новой серии.

На схемах прочих окон надпись с названием приложения поясняться не будет, так как несёт она ту же функцию, равно как и надпись с краткой инструкцией – в дальнейшем она будет содержать название открытого окна.

Схема окна редактирования серии представлена ниже (рисунок 30).



- 1 – окно вывода серии с названиями тонов и их октав;
- 2 – кнопка для начала ввода тонов (редактирования серии);
- 3 – кнопка для начала анализа серии;
- 4 – кнопка для генерации матрицы серии;
- 5 – кнопка для возврата к предыдущему окну.

Рисунок 30 – Схема окна редактирования серии

Для совершения перехода к окну, отвечающему за ввод информации о положении тонов (окно «Ввод тонов») пользователю предоставляется возможность воспользоваться соответствующей кнопкой на представленном на рисунке выше окне.

Схема окна ввода тонов представлена ниже (рисунок 31).

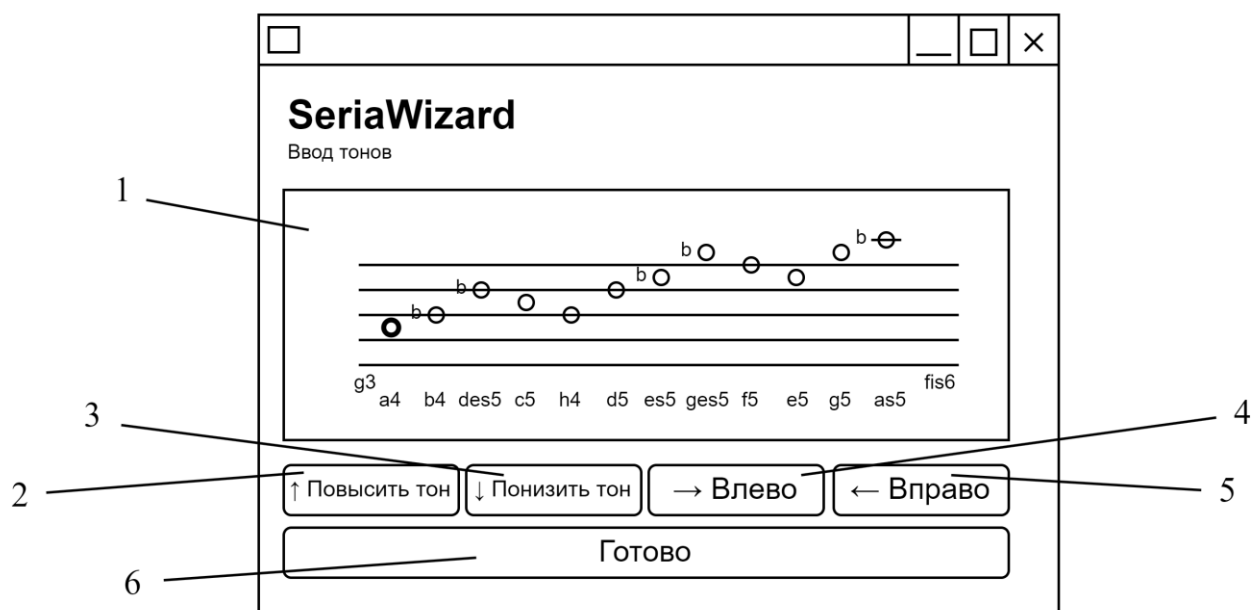


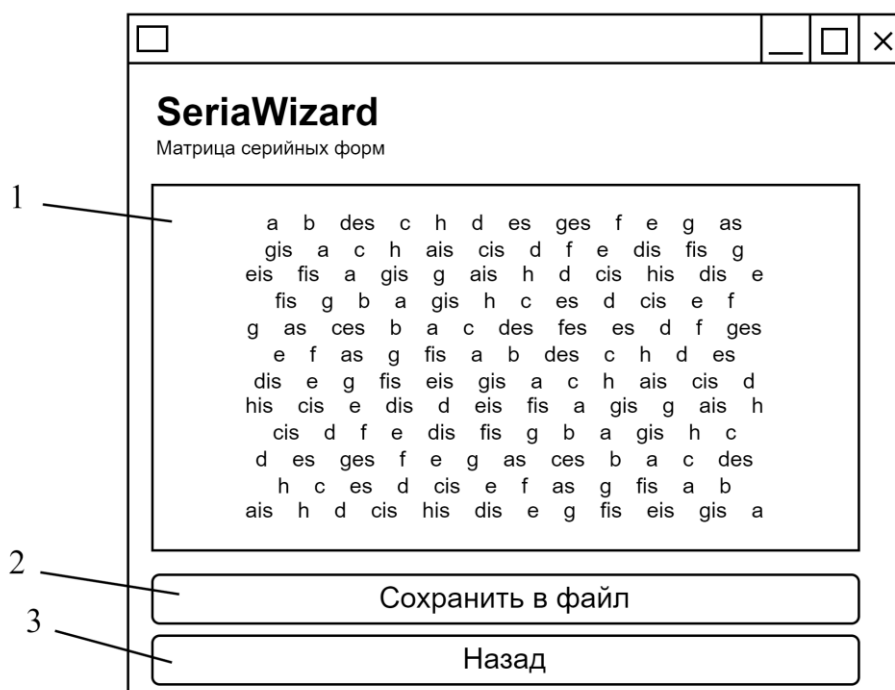
Рисунок 31 – Схема окна ввода тонов

Окно имеет следующие объекты интерфейса:

- 1 – окно вывода серии с названиями тонов и их октав (но с «подсветкой» выделенного в процессе редактирования тона);
- 2 – кнопка для повышения тона;
- 3 – кнопка для понижения тона;
- 4 – кнопка для перехода к следующему тону (влево);
- 5 – кнопка для перехода к предыдущему тону (вправо);
- 6 – кнопка для завершения редактирования и перехода к предыдущему окну.

К окну, предназначенному для вывода матрицы форм двенадцатитоновой серии, можно перейти только после достаточно успешного прохождения анализа введённого ряда. Именно окно вывода является главным с точки зрения выделяемой основной задачи разрабатываемого приложения – автоматизированного построения матрицы форм двенадцатитоновой серии средствами электронно-вычислительной машины.

Схема окна, отвечающего за вывод матрицы, представлена ниже (рисунок 32).



- 1 – окно вывода матрицы на экран;
- 2 – кнопка для сохранения матрицы в файл;
- 3 – кнопка для возврата к предыдущему окну.

Рисунок 32 – Схема окна вывода матрицы форм серии

Так, были разработаны все окна приложения для обеспечения доступа ко всему функционалу разрабатываемого приложения, решающему все поставленные перед разработкой задачи.

4 Программная реализация

Программная реализация проходила в среде разработки Microsoft Visual Studio 2022 на языке C++.

Для более ясного представления статического представления дизайна системы целесообразно использовать диаграмму компонентов. Она демонстрирует инкапсулированные классы и их интерфейсы, порты и внутренние структуры, состоящие из вложенных компонентов и коннекторов [19].

Созданная диаграмма представлена ниже (рисунок 33).

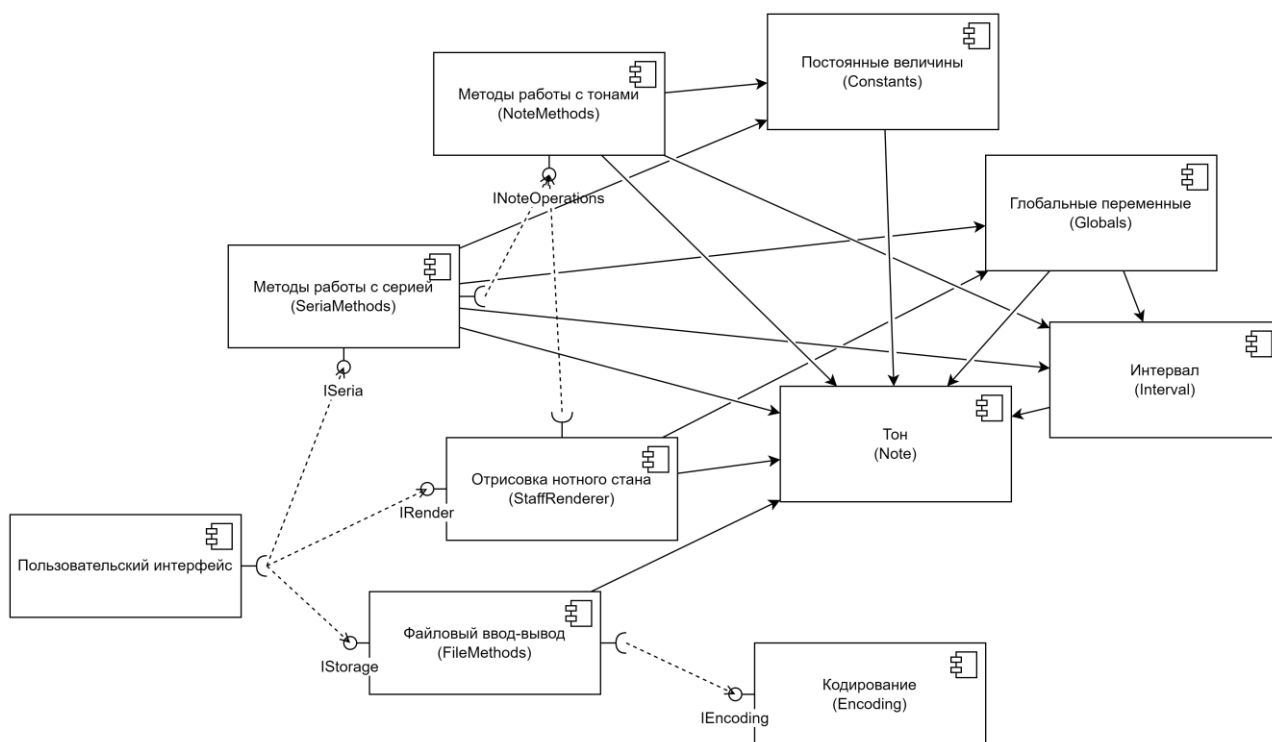


Рисунок 33 – Диаграмма компонентов приложения для генерации матрицы серийных форм «SeriaWizard»

Как можно заметить из представленной диаграммы, важнейшим компонентом в архитектуре приложения является класс, описывающий тон – именно на основе него и функционируют каждый из специализированных алгоритмов.

Для демонстрации декомпозиции модели разрабатываемого приложения на организационные единицы и их зависимости была составлена диаграмма пакетов (рисунок 34).

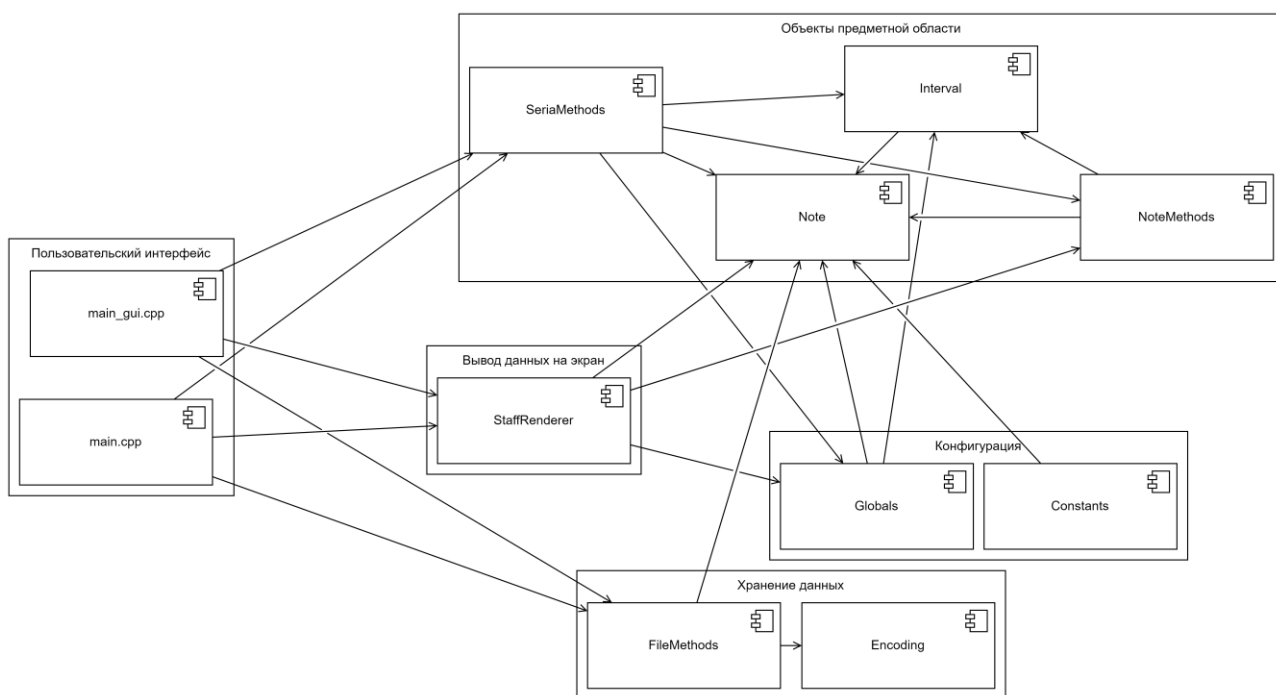


Рисунок 34 – Диаграмма пакетов приложения для генерации матрицы серийных форм «SerialWizard»

Для более наглядного представления физической архитектуры разрабатываемого программного обеспечения была создана диаграмма развёртывания (рисунок 35).

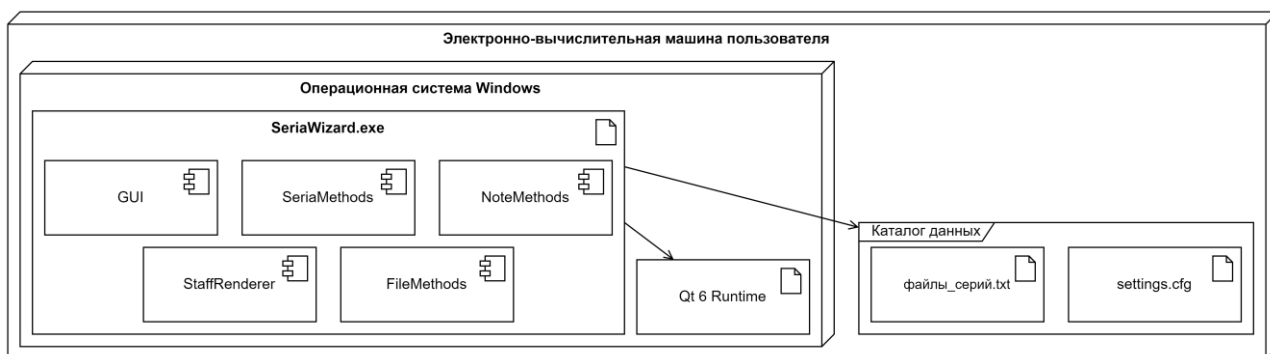


Рисунок 35 – Диаграмма развёртывания приложения для генерации матрицы серийных форм «SerialWizard»

Был написан код реализации метода «primusSeria» (листинг 1).

Листинг 1 – Метод для возврата Primus серии

```
void SerialMethods::primusSeria(const Note &key, const Interval
*primusIntervals, Note *outSeria) {
    outSeria[0] = key;
```

```

        clampIntoSystemRange(outSeria[0]);
        for (int i = 0; i < INTERVAL_COUNT; ++i) {
            outSeria[i + 1] = getNewNoteFromInterval(outSeria[i],
primusIntervals[i]);
            clampIntoSystemRange(outSeria[i + 1]);
        }
    }
}

```

Так, метод принимает начальный тон серии («key»), указатель на массив интервалов серии и возвращает указатель на заполненный массив тонов («outSeria») в соответствии с разработанным ранее алгоритмом (смотри рисунок 23).

Был написан код реализации метода «inversusSeria» (листинг 2).

Листинг 2 – Метод для возврата Inversus серии

```

void SeriaMethods::inversusSeria(const Note *seria, const Interval
*seriaIntervals, Note *outSeria) {
    Interval invIntervals[INTERVAL_COUNT];
    makeInversusIntervals(seriaIntervals, invIntervals);
    primusSeria(seria[0], invIntervals, outSeria);
}

```

Метод принимает на вход указатель на массив с исходной серией («*seria»), указатель на массив интервалов серии («seriaIntervals») и возвращает указатель на заполненный массив тонов («outSeria») в соответствии с разработанным ранее алгоритмом (смотри рисунок 25).

Был написан код реализации метода «buildMatrix» (листинг 3).

Листинг 3 – Метод для возврата матрицы форм серии

```

void SeriaMethods::buildMatrix(const Note *primus, Note
*matrixRowMajor144) {
    Interval invIntervals[INTERVAL_COUNT];
    makeInversusIntervals(thisIntervals, invIntervals);

    for (int c = 0; c < SERIA_COUNT; ++c) {
        Note column[SERIA_COUNT];
        primusSeria(primus[c], invIntervals, column);
        for (int r = 0; r < SERIA_COUNT; ++r) {
            matrixRowMajor144[r * SERIA_COUNT + c] = column[r];
        }
    }

    for (int c = 0; c < SERIA_COUNT; ++c) {
        matrixRowMajor144[c] = primus[c];
    }
}

```

Метод принимает на вход указатель на массив с Primus серии («*primus») и возвращает указатель на массив («*matrixRowMajor144»), являющимся в конце операции матрицей форм. В процессе построения происходит обращение к методам «primusSeria» и «inversusSeria». Матрица строится по следующему принципу: сперва строится инверсия от основного тона исходной формы и записывается в первый столбец, затем то же происходит с каждым из оставшихся тонов. В итоге получается верная матрица форм серии.

Для анализа первого правила серии был создан метод «rule1» (листинг 4).

Листинг 4 – Метод для проверки первого правила построения серии

```
bool SeriaMethods::rule1(const Interval *iv, const Note *s, int n)
{
    if (s && allSemitoneSteps(s, n)) {
        return false;
    }
    bool allMinSeconds = true;
    for (int i = 0; i < n; ++i) {
        if (!(iv[i].getIndex() == 1 && iv[i].getType() == -1)) {
            allMinSeconds = false;
            break;
        }
    }
    if (allMinSeconds) {
        return false;
    }
    bool foundImperfect = false;
    for (int i = 0; i < n; ++i) {
        if (!PERFECT_CONSONANCES.count(iv[i].getIndex())) {
            foundImperfect = true;
            break;
        }
    }
    if (!foundImperfect) {
        return false;
    }
    return true;
}
```

Метод получает на вход указатели на массив интервалов серии («*iv») и на массив тонов серии («*s»), а также количество интервалов («n»). Сперва проводится проверка на следующие друг за другом интервалы малой секунды (проверка на хроматическую гамму), а затем – на чередование чистых кварт и квинт

(проверка на тождественность кварто-квинтовому кругу). В конце возвращается логическое значение в качестве результата анализа по первому правилу.

Для анализа второго правила серии был создан метод «rule2» (листинг 5).

Листинг 5 – Метод для проверки второго правила построения серии

```
bool SeriaMethods::rule2(const Note *s) {
    for (int i = 0; i < SERIA_COUNT; ++i) {
        for (int j = i + 1; j < SERIA_COUNT; ++j) {
            if (s[i] == s[j]) {
                return false;
            }
        }
    }
    return true;
}
```

Метод принимает указатель на массив с серией и проходит по каждому его элементу, сравнивая: повторен ли элемент. Если повторен – возвращает ЛОЖЬ. В противном случае – ИСТИНУ.

Для анализа четвёртого правила серии был создан метод «rule4» (листинг 6).

Листинг 6 – Метод для проверки четвёртого правила построения серии

```
bool SeriaMethods::rule4(const Note *s) {
    Note sorted[SERIA_COUNT];
    for (int i = 0; i < SERIA_COUNT; ++i) {
        sorted[i] = s[i];
    }

    for (int i = 0; i < SERIA_COUNT - 1; ++i) {
        for (int j = 0; j < SERIA_COUNT - i - 1; ++j) {
            if (sorted[j + 1] < sorted[j]) {
                const Note tmp = sorted[j];
                sorted[j] = sorted[j + 1];
                sorted[j + 1] = tmp;
            }
        }
    }

    Interval span;
    span.getInterval(sorted[0], sorted[SERIA_COUNT - 1]);
    if (span.getIndex() >= 9 && span.getType() > 0) {
        return false;
    }
    return true;
}
```

Метод принимает на вход указатель на массив с серией, проходит по каждому элементу и методом пузырька получает самый низкий тон и самый высокий. После прохода определяет интервал между ними (диапазон серии). Если он выходит за интервал большой квартдецимы – правило нарушается, и возвращается ЛОЖЬ. В противном случае (интервал меньше большой квартдецимы) – ИСТИНА.

Для анализа пятого правила серии был создан метод «rule5» (листинг 7).

Листинг 7 – Метод для проверки пятого правила построения серии

```
bool SeriaMethods::rule5(const Interval *iv, int n) {
    for (int i = 0; i < n; ++i) {
        if (iv[i].getIndex() > 7) {
            return false;
        }
    }
    return true;
}
```

Метод принимает на вход указатель на массив интервалов серии и количество интервалов. Проходит циклом по всему массиву и в случае обнаружения интервала, превышающего октаву, возвращает ЛОЖЬ. Соответственно, если все интервалы не превышают октавы, возвращает ИСТИНУ.

Для анализа шестого правила серии был создан метод «rule6» (листинг 8).

Листинг 8 – Метод для проверки шестого правила построения серии

```
bool SeriaMethods::rule6(const Interval *iv, int n) {
    if (n < 3) {
        return true;
    }
    for (int i = 0; i <= n - 3; ++i) {
        if (iv[i] == iv[i + 1] && iv[i + 1] == iv[i + 2]) {
            return false;
        }
    }
    return true;
}
```

Метод так же принимает указатель на массив интервалов и количество их. Возвращает ложь в случае трёх идущих подряд одинаковых интервалов.

Для анализа седьмого правила серии был создан метод «rule7» (листинг 9).

Листинг 9 – Метод для проверки седьмого правила построения серии

```
bool SerialMethods::rule7(const Interval *iv, int n) {
    if (n < 3) {
        return true;
    }
    for (int i = 0; i <= n - 3; ++i) {
        const Interval &a = iv[i];
        const Interval &b = iv[i + 1];
        const Interval &c = iv[i + 2];
        const bool aWide = a.isItWide();
        const bool cWide = c.isItWide();
        const bool aTight = a.getIndex() < 4;
        const bool cTight = c.getIndex() < 4;
        if (aWide && cTight) {
            return false;
        }
        if (aTight && cWide) {
            return false;
        }
        const bool sameKind = (aWide && cWide) || (aTight &&
cTight);
        if (sameKind && b == a) {
            return false;
        }
    }
    return true;
}
```

Этот метод снова работает с интервалами серии. Проходит по всему массиву, беря к рассмотрению по три элемента. Так, ЛОЖЬ будет возвращена в случаях:

- а) широкий, любой, узкий;
- б) узкий, любой, широкий;
- в) узкий, узкий, узкий;
- г) широкий, широкий, широкий.

Соответственно, ИСТИНА – при благоприятном исходе.

Важно обратить внимание на то, в каком виде будут сохраняться данные в текстовых документах. В операциях внутри приложения должна использоваться американская система записи, в то время как вывод будет представлен записью немецкой¹⁵.

¹⁵ Смотри гл. 1.2.2.

Так будет выглядеть запись серии, представленной ранее (смотри рисунок 10), в текстовом файле: «a4 b4 des5 c5 h4 d5 es5 ges5 f5 e5 g5 as5». Здесь учтены немецкая нотация и октавы, в которых располагаются тоны. Таким образом, при необходимости, допускается возможность изменения серии через текстовые редакторы.

Полный программный код разрабатываемого приложения представлен в приложении Б.

5 Контроль качества программного обеспечения и тестирование приложения

5.1 Выбор методов контроля качества и тестирования программного обеспечения

Сергей Куликов даёт следующее определение тестированию: «Тестирование программного обеспечения – процесс анализа программного средства и сопутствующей документации с целью выявления дефектов и повышения качества продукта» [27, стр. 6].

За всё время существования мира информационных технологий сформировалось множество способов тестирования. В рамках данной работы было принято решение обратиться к методу так называемого «чёрного ящика».

Тестирование чёрного ящика или поведенческое тестирование – стратегия (метод) тестирования функционального поведения объекта (программы, системы) с точки зрения внешнего мира, при котором не используется знание о внутреннем устройстве (коде) тестируемого объекта [28].

Для оценки сложности разрабатываемого программного обеспечения было принято решение использовать метрики SLOC и метрики Холстеда.

Количество строк исходного кода (Source Lines of Code – SLOC) является наиболее простым и распространенным способом оценки объёма работ по проекту.

В зависимости от того, каким образом учитывается сходный код, выделяют два основных показателя SLOC:

Количество «физических» строк кода – SLOC (используемые аббревиатуры LOC, SLOC, KLOC, KSLOC, DSLOC) – определяется как общее число строк исходного кода, включая комментарии и пустые строки (при измерении показателя на количество пустых строк, как правило, вводится ограничение – при подсчёте учитывается число пустых строк, которое не превышает 25 % общего числа строк в измеряемом блоке кода).

Количество «логических» строк кода – SLOC (используемые аббревиатуры LSI, DSI, KDSI, где «SI» – source instructions) – определяется как количество команд и зависит от используемого языка программирования. В том случае, если язык не допускает размещение нескольких команд на одной строке, то количество «логических» SLOC будет соответствовать числу «физических», за исключением числа пустых строк и строк комментариев. В том случае, если язык программирования поддерживает размещение нескольких команд на одной строке, то одна физическая строка должна быть учтена как несколько логических, если она содержит более одной команды языка.

Для метрики SLOC существует большое число производных, призванных получить отдельные показатели проекта, основными среди которых являются:

- число пустых строк;
- число строк, содержащих комментарии;
- процент комментариев (отношение строк кода к строкам комментария, производная метрика стилистики);
- среднее число строк для функций (классов, файлов);
- среднее число строк, содержащих исходный код для функций (классов, файлов);
- среднее число строк для модулей [29].

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

- NUOprtr (Number of Unique Operators) – число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
- NUOprnd (Number of Unique Operands) – число уникальных операндов программы (словарь операндов);
- Noprtr (Number of Operators) – общее число операторов в программе;
- Noprnd (Number of Operands) – общее число операндов в программе.

На основании этих характеристик рассчитываются оценки. Формулы их расчёта приведены ниже.

Словарь программы (Halstead Program Vocabulary, HPVoc):

$$HPVoc = NUOprtr + NUOprnd.$$

Длина программы (Halstead Program Length, HPLen):

$$HPLen = Noprtr + Noprnd.$$

Объём программы (Halstead Program Volume, HPVol):

$$HPVol = HPLen \log_2 HPVoc.$$

Сложность программы (Halstead Difficulty, HDiff):

$$HDiff = \frac{NUOprtr}{2} \times \frac{NOprnd}{NUOprnd}.$$

Помимо прочего потребуется определить время работы каждого из основных методов: для анализа серии и для построения матрицы.

Этим инструментами анализа можно дать достаточную оценку качества разрабатываемого приложения.

5.2 Оценка качества кода метриками SLOC

Было принято решение оценить метрики кода из файла «SeriaMethods.cpp» (смотри приложение Б, листинг Б.14), так как именно в нём описаны главные методы для анализа серии и построения матрицы.

Для расчёта строк с исходным кодом была использована формула:

$$N_{с. к.} = N_{строк} - N_{пуст.} - N_{комм.},$$

где $N_{с. к.}$ – количество «логических» строк;

$N_{строк}$ – количество «физических» строк;

$N_{пуст.}$ – количество пустых строк;

$N_{комм.}$ – количество строк-комментариев.

Среднее число строк на метод было получено при использовании формулы:

$$N_{счсчм} = \frac{N_{строк}}{N_{мет.}} = \frac{298}{19} \approx 15,68,$$

где $N_{счсчм}$ – среднее число «физических» строк на метод;

$N_{\text{строк}}$ – количество «физических» строк;

$N_{\text{мет.}}$ – число методов.

Аналогичным способом было высчитано и среднее число строк исходного кода на метод: $N_{\text{счсикнм}}$ равно 14,05.

Цикломатическую сложность было решено определить лишь для наиболее важных методов, непосредственно направленных на решение поставленных перед разрабатываемым приложением задач – метод построения матрицы и методы оценки качества серии.

Цикломатическая сложность высчитывалась с помощью формулы Маккейба:

$$V(G) = P + I,$$

где $V(G)$ – цикломатическая сложность;

P – количество точек принятия решения¹⁶.

Результаты представлены в таблице ниже (таблица 5).

Таблица 5 – Цикломатическая сложность методов класса «SeriaMethods»

№ п/п	Метод	Количество точек принятия решения	Цикломатическая сложность
1	buildMatix()	2 «for»	3
2	rule1()	4 «if», 2 «for»	7
3	rule2()	1 «if», 2 «for»	4
4	rule4()	2 «if», 2 «for»	5
5	rule5()	1 «if», 1 «for»	3
6	rule6()	2 «if», 1 «for»	4
7	rule7()	5 «if», 1 «for»	7

Среднее значение цикломатической сложности было получено с использованием формулы:

$$V(G)_{\text{ср.}} = \frac{\sum_{i=1}^{N_{V(G)}} V(G)_i}{N_{V(G)}} = \frac{3 + 7 + 4 + 5 + 3 + 4 + 7}{7} \approx 4,71,$$

где $V(G)_{\text{ср.}}$ – среднее значение цикломатической сложности;

$V(G)$ – цикломатическая сложность;

$V(G)_i$ – i -я цикломатическая сложность.

¹⁶ Было решено не принимать логические операции в качестве точек принятия решения при подсчёте цикломатической сложности.

Полученные значения не превышают 10, что свидетельствует об умеренной сложности методов.

Все полученные метрики SLOC представлены ниже (таблица 6).

Таблица 6 – Метрики SLOC класса «SeriaMethods»

№ п/п	Метрика	Значение
1	Количество «физических» строк	298
2	Количество пустых строк	31
3	Количество строк-комментариев	0
4	Количество «логических» строк	267
5	Процент строк-комментариев	0
6	Количество методов	19
7	Среднее число строк на метод	15,68
8	Среднее число «логических» строк на метод	14,05
9	Средняя цикломатическая сложность основных методов	4,71
10	Максимальная цикломатическая сложность основных методов	7

Исходя из полученных результатов метрик SLOC можно сделать вывод о достаточно компактном и разборчивом для прочтения другим программистом кодом.

5.3 Оценка качества кода метриками Холстеда

В процессе оценки качества наиболее значимых методов для решения поставленных перед разрабатываемым приложением задач (а именно методов построения матрицы и определения нарушений правил построения двенадцатитоновой серии) были совершены математические операции, направленные на определение значений метрик Холстеда.

Результаты вычислений приведены в таблице ниже (таблица 7).

Таблица 7 – Метрики Холстеда для основных методов класса «SeriaMethods»

№ п/п	Метод	NUOprtr	NUOprnd	Noprtr	Noprnd	HPVoc	HPLen	HPVol	HDiff
1	«buildMatix()»	12	11	24	30	23	54	244	16,4
2	«rule1()»	12	11	33	42	23	75	339	22,9
3	«rule2()»	8	6	13	18	14	31	118	12,0
4	«rule4()»	11	10	28	34	21	62	272	18,7
5	«rule5()»	7	5	8	10	12	18	65	7,0
6	«rule6()»	9	7	15	18	16	33	132	11,6
7	«rule7()»	14	12	38	46	26	84	395	26,8

Наибольшая длинна (HPLen) наблюдается у метода «rule7» – 395 единиц. Это обусловлено сложностью проверки, чередуются ли узкие и широкие интервалы¹⁷. Этим же объясняется и наибольшая сложность (HPDiff), равная 26,8 единицам.

Вторым по сложности стоит метод «rule1», проверяющий, не является ли построение хроматической гаммой.

Напротив, наиболее простой метод – «rule5». Он проверяет, был ли совершён скачок, превышающий интервал октавы.

В истории оценочной практики кода сформировались следующие интерпретации показателя сложности (HDiff):

- HDiff меньше 10 единиц – простая сложность кода;
- HDiff равен 10 – 20 единицам – средняя сложность кода;
- HDiff равен 20 – 30 единицам – высокая сложность кода;
- HDiff больше 30 единиц – очень высокая сложность кода [29].

Ни один из методов по сложности кода не перешёл порог в 30 единиц.

5.4 Функциональное тестирование приложения

Для основных вариантов использования разрабатываемого приложения было выполнено функциональное тестирование основных функций приложения (таблица 8).

Таблица 8 – Результаты функционального тестирования

№ п/п	Описание теста	Входные данные	Ожидаемый результат	Результат теста
1	2	3	4	5
1	Запуск приложения	Запуск исполняемого файла приложения	Приложение запустилось	УСПЕХ

¹⁷ Подробнее об этом было сказано в гл. 4.

Продолжение таблицы 8

1	2	3	4	5
2	Загрузка серии из файла	Файл с серией	Серия успешно загрузилась	УСПЕХ
3	Загрузка набора символов из файла	Файл с набором символов	Вывелась ошибка прочтения файла	УСПЕХ
4	Загрузка серии, записанной с ошибками	Файл с серией, записанной с ошибками	Вывелась ошибка прочтения файла	УСПЕХ
5	Создание новой серии	Нажатие на кнопку «Создать серию» в основном меню приложения	Открылось окно создания серии	УСПЕХ
6	Редактирование серии	Нажатие на кнопку «Ввод тонов» в окне «Редактирование серии» и введение тонов с клавиатуры	Открылось окно «Ввод тонов». Все данные, полученные с клавиатуры, трактованы верно	УСПЕХ
7	Проведение анализа неудачной серии	Нажатие на кнопку «Анализ серии» в окне «Редактирование серии» с загруженной серией, нарушающей правило б)	Вывелся результат анализа и сообщение о невозможности построения матрицы	УСПЕХ
8	Проведение анализа достаточно удачной серии	Нажатие на кнопку «Анализ серии» в окне «Редактирование серии» с загруженной серией, нарушающей правила г), е) и ж)	Вывелся результат анализа и сообщение о возможности построения матрицы	УСПЕХ

Окончание таблицы 8

1	2	3	4	5
9	Проведение анализа ортодоксальной серии	Нажатие на кнопку «Анализ серии» в окне «Редактирование серии» с загруженной ортодоксальной серией	Вывелся результат анализа (ошибок не обнаружено) и сообщение о возможности построения матрицы	УСПЕХ
10	Построение матрицы форм ортодоксальной серии	Нажатие на кнопку «Генерация матрицы» в окне «Редактирование серии» при загруженной серии и достаточно успешно пройденном анализе	Открылось окно «Матрица серийных форм», и вывелась верная матрица форм серии	УСПЕХ
11	Построение матрицы при проваленном анализе	Нажатие на кнопку «Генерация матрицы» в окне «Редактирование серии» с плохим результатом анализа	Вывелось сообщение о недостаточно успешно пройденном анализе.	УСПЕХ
12	Сохранение построенной матрицы в файл	Нажатие на кнопку «Сохранить в файл» в окне «Матрица серийных форм»	Матрица успешно сохранилась	УСПЕХ
13	Получение предупреждения о возможной потере несохранённых данных при создании новой серии	Нажатие на кнопку «Новая серия» в основном меню приложения	Вывелось сообщение с предупреждением о возможной потере несохранённых данных	УСПЕХ
14	Получение информации о приложении	Нажатие на кнопку «Информация» в основном меню приложения	Вывелось сообщение с информацией о приложении	УСПЕХ
15	Получение предупреждения о возможной потере несохранённых данных при выходе из приложения	Нажатие на кнопку «Выход» в основном меню приложения	Вывелось сообщение с предупреждением о возможной потере несохранённых данных при выходе	УСПЕХ

Результаты тестирования представлены на экранных снимках в приложении В.

Для подсчёта времени работы приложения, уходящего на анализ серии и построение матрицы её форм было принято решение воспользоваться

машинным подсчётом путём временного изменения кодов методов, отвечающих за данные задачи.

Так, программный код метода для проведения анализа серии был дополнен для реализации машинного подсчёта времени действия алгоритма (листинг 10).

Листинг 10 – Изменённый код метода для анализа серии

```
void SerialMethods::runAnalysis(std::string &msg, int &code) {
    // Фиксация отправной временной точки
    auto start = std::chrono::high_resolution_clock::now();

    msg.clear();
    const bool r1 = rule1(thisIntervals, thisSerial,
INTERVAL_COUNT);
    if (!r1) {
        msg += u8"\na) серия не должна быть тождественна
хроматической гамме, квартовому или квинтовому кругу";
    }
    const bool r2 = rule2(thisSerial);
    if (!r2) {
        msg += u8"\nb) каждый тон в этом схематическом
образовании появляется только однажды";
    }
    const bool r4 = rule4(thisSerial);
    if (!r4) {
        msg += u8"\nr) серия не должна слишком выходить за рамки
диапазона октавы или ноты (ваша вышла за "
        u8"большую квартдециму)";
    }
    const bool r5 = rule5(thisIntervals, INTERVAL_COUNT);
    if (!r5) {
        msg += u8"\nd) скачки больше чем на октаву, учитывая
возможные октавные транспозиции тонов, не имеют "
        u8"смысла";
    }
    const bool r6 = rule6(thisIntervals, INTERVAL_COUNT);
    if (!r6) {
        msg += u8"\ne) не больше двух одинаковых интервалов могут
следовать подряд друг за другом";
    }
    const bool r7 = rule7(thisIntervals, INTERVAL_COUNT);
    if (!r7) {
        msg += u8"\nj) малые и большие (широкие) интервалы лучше
чередовать";
    }
    const bool r8 = rule8(thisIntervals, INTERVAL_COUNT);
    if (!r8) {
        msg += u8"\nz) не допускаются мелодические ходы,
создающие арпеджио диатонических аккордов (особенно "
        u8"трезвучий)";
    }
}
```

```

    }

    if (!r1 || !r2 || !r5) {
        code = -1;
    } else if (!r4) {
        code = 1;
    } else if (!r6 || !r7 || !r8) {
        code = 2;
    } else {
        code = 3;
    }

    if (code <= 0) {
        msg += u8"\n\nРезультат - провальный. Построить матрицу и
вывести формы невозможно.";
    } else if (code == 1) {
        msg +=
            u8"\n\nРезультат - успешный, однако вывести
формы будет невозможно (слишком широкий диапазон серии).";
    } else if (code == 2) {
        msg += u8"\n\nРезультат - успешный.";
    } else {
        msg += u8"\n\nРезультат - успешный! Серия
ортодоксальная.";
    }

    // Фиксация заключительной временной точки
    auto end = std::chrono::high_resolution_clock::now();
    // Подсчёт продолжительности работы алгоритма
    auto duration =
std::chrono::duration_cast<std::chrono::microseconds>(end -
start);
    // Инициализация переменной для создания сообщения
    std::wstring message = L"TIME = " +
std::to_wstring(duration.count()) + L" μs";
    MessageBoxW(
        NULL,
        message.c_str(),
        L"TIME",
        MB_OK | MB_ICONINFORMATION
    );
}

```

Для подсчёта времени использованы методы из библиотеки «chrono» (в заголовке прописано «`#include <chrono>`»), для вывода результата – заголовочный файл «`windows.h`».

Результаты тестирования времени работы метода для проведения анализа двенадцатитоновой серии представлены в таблице ниже (таблица 9).

Таблица 9 – Результаты анализа времени работы метода для проведения анализа серии

№ п/п	1	2	3	4	5	6	7	8	9	10
Время, мс	17	9	8	13	8	9	9	10	10	8

Для подсчёта среднего значения времени работы метода была применена формула:

$$\text{Среднее время} = \frac{\sum_{i=1}^n x_i}{n},$$

где n – количество полученных результатов;

x_i – значение i -го результата (в мс).

В итоге среднее время работы метода для проведения анализа серии равно 10,1 мс.

Аналогичный порядок действий был применён и к методу, строящему матрицу (листинг 11).

Листинг 11 – Изменённый код метода для построения матрицы форм серии

```
void SeriaMethods::buildMatrix(const Note *primus, Note
*matrixRowMajor144) {
    // Фиксация отправной временной точки
    auto start = std::chrono::high_resolution_clock::now();

    Interval invIntervals[INTERVAL_COUNT];
    makeInversusIntervals(thisIntervals, invIntervals);

    for (int c = 0; c < SERIA_COUNT; ++c) {
        Note column[SERIA_COUNT];
        primusSeria(primus[c], invIntervals, column);
        for (int r = 0; r < SERIA_COUNT; ++r) {
            matrixRowMajor144[r * SERIA_COUNT + c] = column[r];
        }
    }

    for (int c = 0; c < SERIA_COUNT; ++c) {
        matrixRowMajor144[c] = primus[c];
    }

    // Фиксация заключительной временной точки
    auto end = std::chrono::high_resolution_clock::now();
    // Подсчёт продолжительности работы алгоритма
    auto duration =
std::chrono::duration_cast<std::chrono::microseconds>(end -
start);
```

```

// Инициализация переменной для создания сообщения
std::wstring message = L"TIME = " +
std::to_wstring(duration.count()) + L" μs";
MessageBoxW(
    NULL,
    message.c_str(),
    L"TIME",
    MB_OK | MB_ICONINFORMATION
);
}

```

Результаты тестирования также представлены в таблице (таблица 10).

Таблица 10 – Результаты анализа времени работы метода для построения матрицы форм серии

№ п/п	1	2	3	4	5	6	7	8	9	10
Время, мс	22	21	20	27	24	35	23	24	21	29

Среднее время работы метода для построения матрицы форм серии равно 24,6 мс.

Для подсчёта, на сколько ускоряются процессы анализа и построения матрицы форм серии, было принято решение использовать формулу:

$$\text{Ускорение процесса} = \frac{T_{\text{старое}} - T_{\text{новое}}}{T_{\text{старое}}} \times 100 \, \%.$$

Так, время, затрачиваемое на анализ серии, сократилось на 99,9976 %, на построение матрицы форм серии – на 99,9992 %¹⁸.

Помимо прочего было совершено тестирование приложения при работе с рядом различных двенадцатитоновых серий.

Ниже представлена первая тестовая серия (рисунок 36).

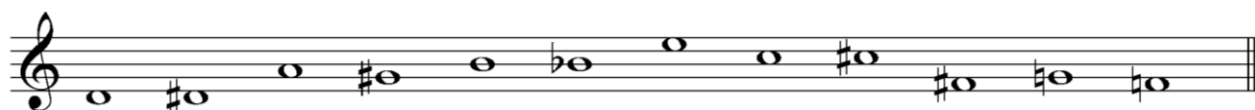


Рисунок 36 – Тестовая двенадцатитоновая серия № 1¹⁹

Анализ должен показать, что серия ортодоксальная.

¹⁸ В качестве старого времени взяты следующие значения: среднее время, затрачиваемое на анализ вручную, равно 7 минутам (или 420.000 миллисекундам); на построение матрицы форм серии – 50 минутам (3.000.000 миллисекунд).

¹⁹ Серия взята из учебного пособия Ц. Когоутека (пример 111, страница 127) [7].

Результат анализа дан на рисунке ниже (рисунок 37).

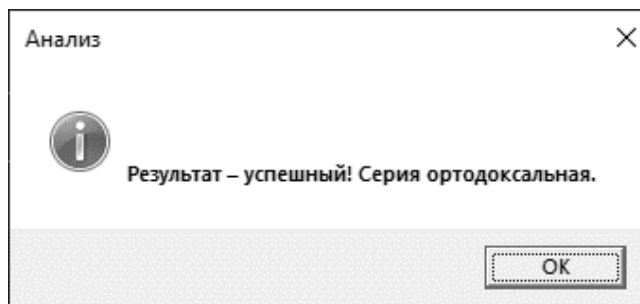


Рисунок 37 – Результат анализа тестовой серии № 1

Матрица тоже построена верно (рисунок 38).

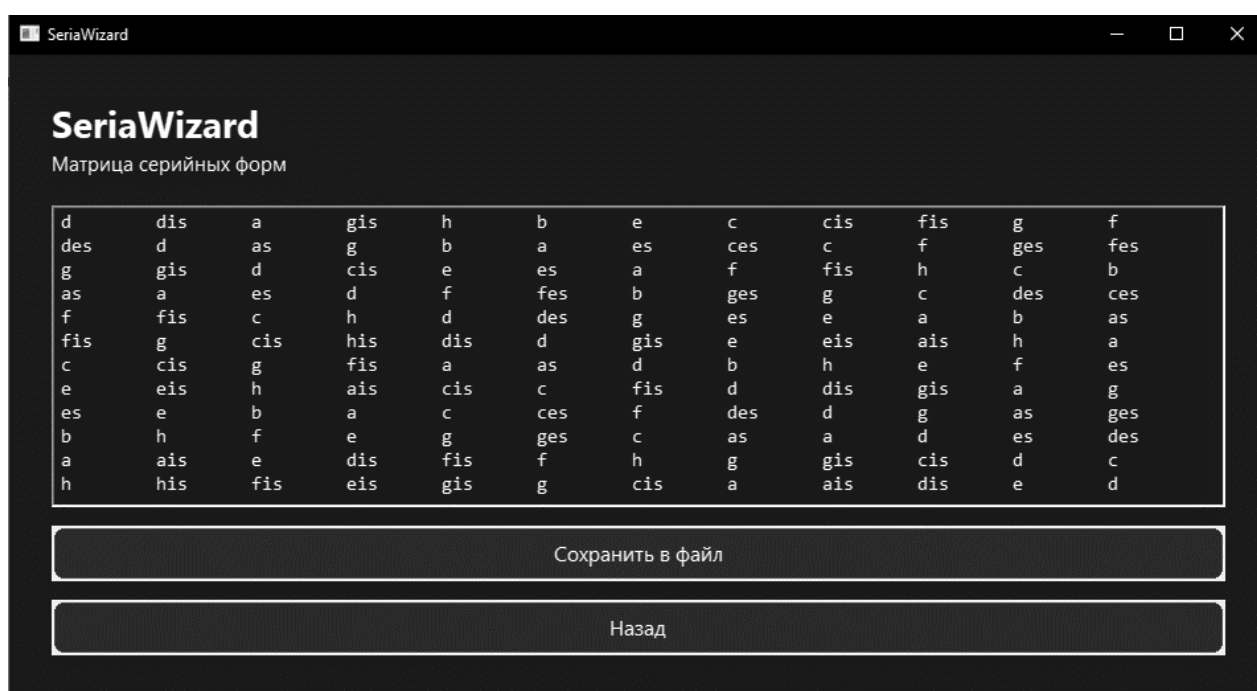


Рисунок 38 – Результат построения матрицы форм тестовой серии № 1

Следующие серии были подобраны так, чтобы они нарушали какие-либо правила (кроме первых двух, разумеется).

Ниже представлена вторая тестовая серия (рисунок 39).



Рисунок 39 – Тестовая двенадцатитоновая серия № 2²⁰

²⁰ Серия взята из учебного пособия Ц. Когоутека (пример 105, страница 124) [7].

Анализ должен показать следующее:

- а) серия слишком выходит за диапазон октавы или ноны;
- б) в серии более двух одинаковых интервалов следуют подряд друг за другом;
- в) не чередуются узкие и широкие интервалы.

Ниже представлен результат анализа (рисунок 40).

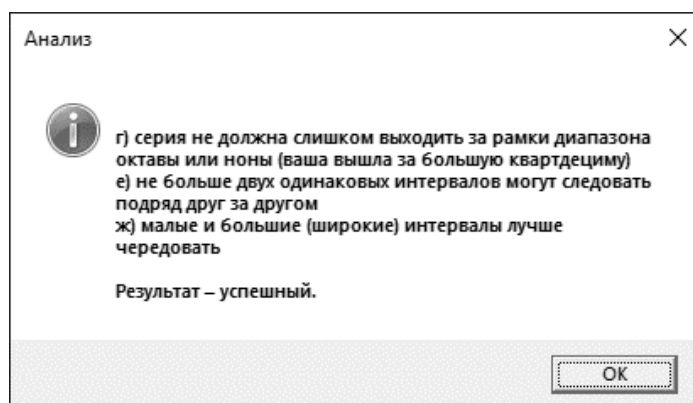


Рисунок 40 – Результат анализа тестовой серии № 2

Построенная матрица представлена ниже (рисунок 41).

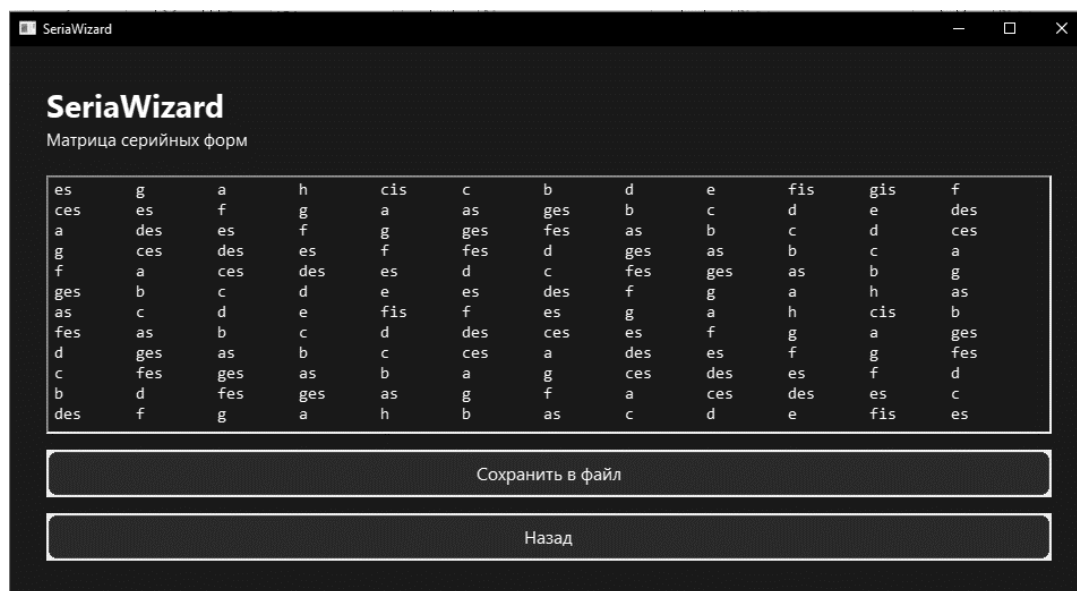


Рисунок 41 – Результат построения матрицы форм тестовой серии № 2

Ниже представлена третья тестовая серия²¹ (рисунок 42).

²¹ Серия взята из учебного пособия Ц. Когоутека (пример 108, страница 125) [7].



Рисунок 42 – Тестовая двенадцатитоновая серия № 3

Анализ должен показать лишь то, что диапазон серии слишком велик.

Результат анализа дан на рисунке ниже (рисунок 43).



Рисунок 43 – Результат анализа тестовой серии № 3

Тестирование показало, что разработанное приложение выполняет поставленные перед ним задачи исправно и достаточно оперативно.

6 Планирование разработки и оценка бюджета

Для планирования разработки проекта и оценки его бюджета была задействована программа ProjectLibre версии 1.9.3. Это кроссплатформенное программное обеспечение для управления проектами. Распространяется на условиях лицензии Common Public Attribution License Version 1.0 (CPAL). Позиционируется создателями как открытая замена коммерческому продукту Microsoft Project. Обеспечение доступно для операционных систем Microsoft Windows, Linux, Unix, Mac OS X [30].

В результате планирования была выделена 21 задача (рисунок 44).

	⑩	Название	Продолжительность	Начало	Окончание	Предшествующие	Название ресурса
1		Получение технического задания на работу	0 дней	26.12.25 8:00	26.12.25 8:00		Аналитик
2		Анализ предметной области и требований к системе	35 дней	07.04.26 8:00	25.05.26 17:00		Аналитик
3		Определение специфики предметной области	5 дней	07.04.26 8:00	13.04.26 17:00		
4		Обзор традиционных графических средств нотации	1 день	07.04.26 8:00	07.04.26 17:00	1	Аналитик
5		Анализ понятий элементарной теории музыки	1 день	08.04.26 8:00	08.04.26 17:00	1	Аналитик
6		Анализ понятий серии и матрицы её форм	3 дней	09.04.26 8:00	13.04.26 17:00		Аналитик
7		Постановка задачи и анализ требований	31 дней	11.04.26 8:00	25.05.26 17:00		
8		Постановка задачи	2 дней	14.04.26 8:00	15.04.26 17:00		Аналитик
9		Анализ систем аналогичного назначения	2 дней	11.04.26 8:00	14.04.26 17:00		Аналитик
10		Формулировка требований к разрабатываемому приложению	1 день	18.04.26 8:00	20.04.26 17:00		Аналитик
11		Выбор и обоснование технологии	25 дней	21.04.26 8:00	25.05.26 17:00		Разработчик[25%];Аналитик
12		Проектирование программных средств	11 дней	27.04.26 8:00	11.05.26 17:00		
13		Проектирование структуры данных	3 дней	27.04.26 8:00	29.04.26 17:00		Разработчик;Аналитик
14		Проектирование алгоритмов	4 дней	01.05.26 8:00	06.05.26 17:00		Разработчик;Аналитик[25%]
15		Проектирование интерфейса	3 дней	07.05.26 8:00	11.05.26 17:00		Разработчик;Аналитик[5%]
16		Реализация программных средств	9 дней	12.05.26 8:00	22.05.26 17:00		Разработчик
17		Тестирование программных средств	8 дней	12.05.26 8:00	21.05.26 17:00		
18		Выбор методов контроля качества и тестирования программного обеспечения	3 дней	12.05.26 8:00	14.05.26 17:00		Разработчик[50%];Аналитик[50%]
19		Оценка качества кода метриками SLOC и Холстеда	1 день	15.05.26 8:00	15.05.26 17:00		Аналитик
20		Функциональное тестирование приложения	4 дней	16.05.26 8:00	21.05.26 17:00		Разработчик[25%];Аналитик
21		Оформление пояснительной записки	8 дней	22.05.26 8:00	02.06.26 17:00		Аналитик

Рисунок 44 – Список задач на проектирование

Для работы над проектом был определён список ресурсов, состоящий из двух сотрудников:

- разработчик;
- аналитик.

Представление дано на рисунке ниже (рисунок 45).

	⑩	Название	Тип	Инициалы	Максимальные единицы	Стандартная ставка	Ставка сверхурочных	Затраты на одно использование	Начислять	Основной календарь
1		Разработчик (Middle)	Работа Разр.		100%	2000 руб./час	2500 руб./час	0 руб.	Пропорционально	Пятидневка
2		Аналитик	Работа Анал.		100%	800 руб./час	900 руб./час	0 руб.	Пропорционально	Пятидневка

Рисунок 45 – Список ресурсов проекта

Для более наглядного представления планирования проекта была построена диаграмма Ганта, помещённая в приложение Д.

Также были подсчитаны трудозатраты часов, отражающие количество времени, затраченное каждым сотрудником на выполнение проекта, и затраты финансовые (рисунок 46).

	Название	Работа	Стоимость
1	Разработчик	198 часов	396000 руб.
	<i>Проектирование структуры данных</i>	<i>24 часов</i>	<i>48000 руб.</i>
	<i>Проектирование интерфейса</i>	<i>24 часов</i>	<i>48000 руб.</i>
	<i>Проектирование алгоритмов</i>	<i>8 часов</i>	<i>16000 руб.</i>
	<i>Функциональное тестирование приложения</i>	<i>8 часов</i>	<i>16000 руб.</i>
	<i>Реализация программных средств</i>	<i>72 часов</i>	<i>144000 руб.</i>
	<i>Выбор методов контроля качества и тестирования программного обеспечения</i>	<i>12 часов</i>	<i>24000 руб.</i>
	<i>Выбор и обоснование технологии</i>	<i>50 часов</i>	<i>100000 руб.</i>
2	Аналитик	445,2 часов	356160 руб.
	<i>Анализ систем аналогичного назначения</i>	<i>16 часов</i>	<i>12800 руб.</i>
	<i>Выбор и обоснование технологии</i>	<i>200 часов</i>	<i>160000 руб.</i>
	<i>Оценка качества кода метриками SLOC и Холстеда</i>	<i>8 часов</i>	<i>6400 руб.</i>
	<i>Анализ понятий серии и матрицы её форм</i>	<i>24 часов</i>	<i>19200 руб.</i>
	<i>Анализ понятий элементарной теории музыки</i>	<i>8 часов</i>	<i>6400 руб.</i>
	<i>Выбор методов контроля качества и тестирования программного обеспечения</i>	<i>12 часов</i>	<i>9600 руб.</i>
	<i>Обзор традиционных графических средств нотописа</i>	<i>8 часов</i>	<i>6400 руб.</i>
	<i>Анализ предметной области и требований к системе</i>	<i>40 часов</i>	<i>32000 руб.</i>
	<i>Оформление пояснительной записки</i>	<i>64 часов</i>	<i>51200 руб.</i>
	<i>Получение технического задания на работу</i>	<i>0 часов</i>	<i>0 руб.</i>
	<i>Постановка задачи</i>	<i>16 часов</i>	<i>12800 руб.</i>
	<i>Проектирование структуры данных</i>	<i>24 часов</i>	<i>19200 руб.</i>
	<i>Формулировка требований к разрабатываемому приложению</i>	<i>8 часов</i>	<i>6400 руб.</i>
	<i>Проектирование интерфейса</i>	<i>1,2 часов</i>	<i>960 руб.</i>
	<i>Функциональное тестирование приложения</i>	<i>8 часов</i>	<i>6400 руб.</i>
	<i>Проектирование алгоритмов</i>	<i>8 часов</i>	<i>6400 руб.</i>

Рисунок 46 – Часы работы каждого сотрудника

Общее количество финансов, направленных на оплату труда разработчика составило 396.000,00 Р.

Общее количество финансов, направленных на оплату труда аналитика составило 356.150,00 Р.

Из предоставленных данных выявлено, что финансовые затраты на разработку проекта составили 752.000,00 Р, трудозатраты – 445,2 человеко-часов. Срок выполнения проекта составил 56 дней.

Заключение

В результате работы было создано консольное приложение «SeriaWizard», позволяющее строить матрицу форм двенадцатитоновых серий, а также проводить достаточный для данной операции анализ серии.

Было проведено освещение элементарных понятий музыкальной теории, а также определение двенадцатитоновой серии. Выделены особенности объектов предметной области.

На этапе анализа требований к разрабатываемому приложению был проведён анализ существующих аналогичных решений: Sibelius, MuseScore и ToneMatrix. Это помогло определиться с оптимальным набором функций, а именно:

- а) ввод двенадцатитоновой серии:
 - 1) с клавиатуры;
 - 2) загрузкой из файла;
- б) редактирование серии;
- в) вывод серии на нотный стан;
- г) анализ серии;
- д) сохранение серии в текстовом файле;
- е) построение матрицы форм серии.

Для реализации был выбран язык C++. В качестве среды разработки – Microsoft Visual Studio 2022.

Особое внимание было уделено выделению объектов предметной области в классы (тон, интервал); алгоритмам, реализующим проверку на предмет несоответствия правилам построения двенадцатитоновой серии, а также алгоритму, реализующему непосредственное построение матрицы форм серии.

Тестирование показало хорошую работу разработанного приложения и достаточное качество кода. Программный код вышел располагающим к расширению.

На последних этапах работы было проведено планирование разработки и оценки бюджета, составлена пояснительная записка в соответствии с требованиями ГОСТ [31 – 35].

Разработанное приложение будет использоваться на отделении «Теория музыки» в Государственном бюджетном профессиональном образовательном учреждении «Пензенский музыкальный колледж имени А. А. Архангельского» для оптимизации работы педагогов на занятиях по дисциплинам «Композиция» и «Современная гармония».

Копия акта о внедрении разработки помещена в приложение Г.

Список использованных источников

1. Энгельс, Ф. Анти-Дюринг / Ф. Энгельс. – М. : Политиздат, 1977. – 430 с.
2. Способин, И. В. Элементарная теория музыки / И. В. Способин, В. Григоренко, О. Кузнецова, Н. Автономов. – М. : Кифара, 1996. – 208 с.
3. Музыкальная энциклопедия. В 6 т. Т. 5. Симон – Хейлер / ред. Ю. В. Келдыш. – М. : Советская энциклопедия, 1981. – 1056 с.
4. Барбур, Дж. М. Настройка и темперация: исторический обзор / Дж. М. Барбур. – Ист-Лансинг : Минеола, 1951. – 228 с.
5. Музыкальный словарь / сост. Г. Риман – М. : Директмедиа Паблишинг, 2008. – с. 10000
6. Янг, Р. В. Терминология для логарифмических единиц частоты / Р. В. Янг // Журнал акустического общества Америки. – 1939. – №11. – С. 134 – 139
7. Когоутек, Ц. Техника композиции в музыке XX века / Ц. Когоутек, пер. К. Н. Иванов – М. : Музыка, 1976. – 368 с.
8. Большая советская энциклопедия. В 30 т. Т. 28 Франкфурт – Чага / гл. ред. А. М. Прохоров. – 3-е изд. – М. : Советская энциклопедия, 1969 – 1978. – 616 с.
9. Шёнберг, А. Стилъ и идея / А. Шёнберг, ред. Л. Стайн, пер. Л. Блэк. – Беркли и Лос-Анджелес : Издательство Калифорнийского университета, 1975. – 567 с.
10. Веберн, А. Путь к новой музыке : лекции, статьи, письма / А. Веберн, пер. с нем. В. Г. Шнитке. – Москва : Музыка, 1975. – 144 с.
11. Гуляницкая, Н. С. Введение в современную гармонию / Н. С. Гуляницкая, Т. С. Бершадская, Ю. Н. Холопов. – М. : Музыка, 1984. – 256 с.
12. Сайт американской технологической компании Avid Technology : официальный сайт [Электронный ресурс]. URL: <https://www.avid.com/sibelius> (дата обращения 24.04.2026)

13. Сапутра, Д. Н. Повышение компетенции студентов в области музыкальной композиции с помощью приложения Sibelius / Д. Н. Сапутра // Авангард: Научный журнал по образованию в сфере исполнительских искусств. – 2020. – №6. – С. 142 – 162
14. Чен, Л. Х. Сравнение программного обеспечения для создания музыки: данные из Sibelius, Cubase и Garage Band / Л. Х. Чен // Прикладная и вычислительная инженерия. – 2025. – №160. – С. 37 – 44
15. Сайт кипрской технологической компании MuseCY Holdings Ltd. : официальный сайт [Электронный ресурс]. URL: <https://musescore.com/> (дата обращения 24.04.2026)
16. Хентшель, Й. Парсер для файлов MuseScore, служащий фабрикой данных для аннотированных музыкальных корпусов / Й. Хейнштель, М. Рормайер // Журнал программного обеспечения с открытым исходным кодом. – 2023. – № 88. – С. 1 – 4
17. Майерс, Б. Краткая история технологий взаимодействия человека с компьютером / Б. Майерс // ACM interactions. – 1998. – №5. – С. 44 – 54
18. Source Forge веб-платформа для размещения, совместной разработки и скачивания программного обеспечения с открытым программным кодом : официальный сайт [Электронный ресурс]. URL: <https://sourceforge.net/projects/tonematrix/> (дата обращения 24.04.2026)
19. Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, И. Якобсон, пер. Н. Мухин. – 2-е изд. – М. : ДМК Пресс. – 496 с.
20. Танеев, С. И. Подвижной контрапункт строгого письма / С. И. Танеев. – М. : М. П. Беляев в Лейпциге, 1909. – 416 с.
21. Доля рынка операционных систем в России // StatCounter Global Stats [Электронный ресурс]. URL: <https://gs.statcounter.com/os-market-share/desktop/russia/> (дата обращения: 20.04.2026)
22. Тернер, Р. Командная строка Windows: Windows-консоль изнутри // Блоги разработчиков Microsoft [Электронный ресурс].

URL: <https://devblogs.microsoft.com/commandline/windows-command-line-inside-the-windows-console/> (дата обращения: 20.04.2026)

23. Рассохин, Д. Язык программирования C++ в хемоинформатике и вычислительной химии / Д. Рассохин // Журнал хемоинформатики. – 2020. – №12. – С. 1 – 24

24. Хьюз, О. Язык программирования C++: как он стал невидимой основой всего // TechRepublic [Электронный ресурс]. – URL: <https://www.techrepublic.com/article/c-programming-language-how-it-became-the-invisible-foundation-for-everything-and-whats-next/> (дата обращения: 20.04.2026)

25. Вайрфреймы (Wireframes) // UsabilityLab [Электронный ресурс]. – URL: <https://usabilitylab.ru/uxmethods/designmethods/wireframes/> (дата обращения: 27.05.2026)

26. UX и UI // ELMA 365 Экосистема Low-code продуктов для автоматизации бизнеса. [Электронный ресурс]. – URL: <https://elma365.com/ru/bazaznaniy/ui-ux/> (дата обращения: 27.05.2026)

27. Куликов, С. Тестирование программного обеспечения. Базовый курс // Хроники детерминированности. [Электронный ресурс]. – URL: https://svyatoslav.biz/software_testing_book/ (дата обращения: 29.05.2026)

28. Бейзер, Б. Тестирование чёрного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер. – СПб : Питер, 2004. – 320 с.

29. Новичков, А. Метрики кода и их практическая реализация в Subversion и ClearCase. Часть 1 – метрики / А. Новичков, А. Шамрай, А. Черников // CM Consult. Консалтинг в области разработки ПО. [Электронный ресурс]. – URL: https://web.archive.org/web/20081018181130/http://cmcons.com/articles/CC_CQ/dev_metrics/mertics_part_1/#1 (дата обращения: 29.05.2026)

30. ProjectLibre : официальный сайт. – Сан-Матео, 2009. – URL: <https://projectlibre.com/> (дата обращения: 10.06.2026)

31. ГОСТ 7.32–2017. Система стандартов по информации, библиотечному и издательскому делу. Отчет о научно-исследовательской работе. Структура и правила оформления. – Введ. 2018–07–01. – М. : Стандартинформ, 2018. – 35 с.

32. ГОСТ Р 2.105–2019. Единая система конструкторской документации. Общие требования к текстовым документам. – Введ. 2021–02–01. – М. : Стандартинформ, 2020. – 40 с.

33. ГОСТ Р 7.0.5–2008. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая ссылка. Общие требования и правила составления. – Введ. 2009–01–01. – М. : Стандартинформ, 2008. – 23 с.

34. ГОСТ 7.1–2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – Введ. 2004–07–01. – Москва : ИПК Издательство стандартов, 2004. – 166 с.

35. ГОСТ Р 7.0.100–2018. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – Введ. 2019–07–01. – М. : Стандартинформ, 2018. – 124 с.

Приложение А
(справочное)
Глоссарий

C++ – это универсальный, высокопроизводительный язык программирования со статической типизацией.

Inversus (инверсия) – обращение музыкального материала.

Muse Score Studio – бесплатный WYSIWYG-редактор с возможностью быстрого ввода нот как с клавиатуры компьютера, так и с внешней MIDI-клавиатуры.

Primus – основной вид музыкального материала.

Retroinversus (ракоход-инверсия) – либо переписанная задом наперёд инверсия, либо инверсия ракохода.

Retroversus (ракоход) – переписанный задом наперёд музыкальный материал.

Sibelius – редактор нотных партитур для Microsoft Windows, Mac OS, Mac OS X и RISC OS от компании Sibelius Software (ныне часть Avid Technology).

SLOC (Source Lines of Code) – это базовая метрика, используемая для оценки объёма программного обеспечения, измерения его сложности и прогнозирования трудозатрат на разработку.

ToneMatrix – любительское оконное приложение за авторством fonazza-stent, пользователя веб-платформы для размещения, совместной разработки и скачивания программного обеспечения с открытым программным кодом Source Forge.

UML (Unified Modeling Language) – унифицированный язык моделирования, являющийся графическим языком для визуализации, специфицирования, конструирования и документирования систем, в которых большая роль принадлежит программному обеспечению.

WYSIWYG (от англ. «What You See Is What You Get» – «что видишь, то и получишь») – свойство прикладных программ или веб-интерфейсов, в которых содержание отображается в процессе редактирования и выглядит максимально близко похожим на конечную продукцию, которая может быть печатным документом, веб-страницей или презентацией.

Аккорд – это одновременное звучание трёх или более музыкальных звуков разной высоты, которые воспринимаются слухом как единое целое.

Алгоритм – это чёткий набор пошаговых инструкций, описывающий порядок действий исполнителя для решения конкретной задачи или достижения определённой цели.

Арпеджио – это способ исполнения аккорда, при котором его звуки берутся не одновременно, а последовательно один за другим.

Бекар – это музыкальный знак альтерации, который отменяет действие ранее выставленных знаков альтерации и возвращает ноте её первоначальное «чистое» звучание.

Бемоль – это музыкальный знак альтерации, который указывает на понижение высоты стоящей рядом ноты на полтона.

Блок-схема – это графическое представление алгоритма или процесса, где каждый этап изображен в виде геометрических фигур (блоков), соединённых стрелками, которые задают последовательность выполнения действий.

Гамма – это последовательность всех звуков лада, расположенных по порядку вверх или вниз в пределах одной или нескольких октав.

Диаграмма классов – это структурная схема в языке UML, которая показывает статическую структуру объектно-ориентированной системы.

Диаграммы вариантов использования – основной вид диаграмм при моделировании поведения системы, подсистемы или класса. Каждая из них показывает набор вариантов использования и действующих лиц в их взаимодействии.

Диатоника – это музыкальная семиступенная система, звуки которой соотносятся друг с другом по чистым квинтам и образуют устойчивый звукоряд.

Диез – это знак альтерации, который повышает высоту ноты на один полтон.

Добавочные линейки на нотоносце – короткие линии для одной ноты, помещаемые выше или ниже нотного стана.

Додекафония – это метод музыкальной композиции, в котором все 12 нот хроматической гаммы уравнены в правах.

Звукоряд – это упорядоченная по высоте последовательность музыкальных звуков (от низких к высоким или наоборот).

Знак альтерации – специальный символ, который указывает на повышение или понижение музыкального звука (ступени) на полтона или целый тон без изменения его названия.

Интервал – это расстояние между двумя звуками по их высоте.

Интерфейс – это совокупность правил, инструментов и средств, которые позволяют разным системам, устройствам или человеку взаимодействовать друг с другом.

Кварто-квинтовый круг – это графическая схема в виде циферблата, которая упорядочивает все 12 тональностей.

Класс – это описание множества объектов с одинаковыми атрибутами, операциями, связями и семантикой.

Лад – это система взаимосвязи музыкальных звуков, которая объединяет их вокруг главного, самого устойчивого звука.

Матрица серийных форм (называемая ещё «волшебным квадратом») – это матрица, горизонталь которой представляет Primus и Retroinversus, а вертикаль – Inversus и Retroinversus.

Метод (в C++) – это функция, объявленная внутри класса или структуры, которая выполняет действия над данными этого класса.

Метрики Холстеда – это набор аналитических показателей в программировании, которые измеряют сложность, объём и усилия, необходимые для понимания или реализации исходного кода

Музыкальный ключ – это специальный знак в начале нотного стана, который задает звуковысотное значение нот.

Музыкальный строй – это система звуковысотных соотношений, которая определяет точную частоту (высоту) каждой ноты в звукоряде.

Нота – специальный символ, обозначающий высоту и длительность музыкального звука.

Нотный стан (или нотоносец) – это система из пяти параллельных горизонтальных линий, на которых, а также между которыми, записываются музыкальные ноты и другие знаки.

Октава – это:

- а) восьмая ступень звукоряда;
- б) регистры равномерно темперированного звукоряда;
- в) интервал, равный расстоянию в восемь ступеней.

Серийная техника – это метод сочинения музыки, использующий только высотные серии.

Серия – это ряд звуков, из которого путём его повторения выводится вся высотная ткань произведения.

Ступень – это отдельный звук в системе звукоряда или гаммы, определяющий его высоту.

Тон – один из компонентов музыкального звука; характеризуется высотой, громкостью и тембром.

Файл – абстракция операционной системы, позволяющая представить какой-либо набор семантически связанных данных как единое целое.

Хроматическая гамма – это музыкальная гамма, состоящая из 12 ступеней, расположенных на расстоянии полутона друг от друга.

Энгармонизм – это полное совпадение нот по высоте и звучанию, но при различном названии и написании на нотном стане.

Приложение Б
(обязательное)
Код программы

Листинг Б.1 – Код «main.cpp»

```

#include "Constants.h"
#include "Encoding.h"
#include "FileMethods.h"
#include "Globals.h"
#include "SeriaMethods.h"

#ifdef NOMINMAX
#define NOMINMAX
#endif
#include <conio.h>
#include <iomanip>
#include <iostream>
#include <string>
#include <vector>
#include <windows.h>

namespace {

    HANDLE conOut() { return GetStdHandle(STD_OUTPUT_HANDLE); }

    void clrScr() {
        CONSOLE_SCREEN_BUFFER_INFO si{};
        COORD origin{0, 0};
        HANDLE h = conOut();
        if (!GetConsoleScreenBufferInfo(h, &si)) {
            return;
        }
        const DWORD cells = static_cast<DWORD>(si.dwSize.X) *
static_cast<DWORD>(si.dwSize.Y);
        DWORD w = 0;
        FillConsoleOutputCharacterA(h, ' ', cells, origin, &w);
        SetConsoleCursorPosition(h, origin);
    }

    void setColor(WORD fg, WORD bg) {
        SetConsoleTextAttribute(conOut(), static_cast<WORD>(fg |
(bg << 4)));
    }

    int readNavKey() {
        int c = _getch();
        if (c == 0 || c == 224) {
            return _getch();
        }
        return c;
    }

    void drawVerticalMenu(const std::string &title, const
std::vector<std::string> &items, int selected,
        const std::string &statusLine, WORD statusAttr,
const std::string &hint) {

```

```

        clrScr();
        setColor(FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE | FOREGROUND_INTENSITY, 0);
        std::cout << title << "\n\n";
        SetConsoleTextAttribute(conOut(), statusAttr);
        std::cout << statusLine << "\n\n";
        setColor(FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE, 0);
        for (int i = 0; i < static_cast<int>(items.size()); ++i)
        {
            if (i == selected) {
                setColor(FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_INTENSITY, 0);
            } else {
                setColor(FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE, 0);
            }
            std::cout << '[' << items[i] << "]\n";
        }
        setColor(FOREGROUND_BLUE | FOREGROUND_INTENSITY, 0);
        std::cout << "\n" << hint << "\n";
    }

    int runVerticalMenu(const std::string &title, const
std::vector<std::string> &items,
        const std::string &statusLine, WORD statusAttr,
const std::string &hint) {
        int sel = 0;
        const int n = static_cast<int>(items.size());
        for (;;) {
            drawVerticalMenu(title, items, sel, statusLine,
statusAttr, hint);
            const int k = readNavKey();
            if (k == 72) {
                sel = (sel - 1 + n) % n;
            }
            if (k == 80) {
                sel = (sel + 1) % n;
            }
            if (k == '\r' || k == 13) {
                return sel;
            }
        }
    }

    void drawHorizontalRow(const std::vector<std::string> &items,
int selected, const std::string &hint) {
        for (int i = 0; i < static_cast<int>(items.size()); ++i)
        {
            if (i > 0) {
                std::cout << " ";
            }
            if (i == selected) {

```

```

        setColor(FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_INTENSITY, 0);
    } else {
        setColor(FOREGROUND_RED | FOREGROUND_GREEN |
FOREGROUND_BLUE, 0);
    }
    std::cout << '[' << items[i] << ']'<
    }
    std::cout << "\n\n";
    setColor(FOREGROUND_BLUE | FOREGROUND_INTENSITY, 0);
    std::cout << hint << "\n";
}

int runHorizontalMenu(const std::vector<std::string> &items,
const std::string &hint) {
    int sel = 0;
    const int n = static_cast<int>(items.size());
    for (;;) {
        clrScr();
        drawHorizontalRow(items, sel, hint);
        const int k = readNavKey();
        if (k == 75) {
            sel = (sel - 1 + n) % n;
        }
        if (k == 77) {
            sel = (sel + 1) % n;
        }
        if (k == '\r' || k == 13) {
            return sel;
        }
    }
}

}

int main() {
    initEncodingConsole();
    const std::string appDir = applicationDirectory();
    resetSeriaState();
    for (;;) {
        if (beginningLoop(appDir) != 0) {
            break;
        }
    }
    clrScr();
    return 0;
}

```

Листинг Б.2 – Код «main_gui.cpp»

```

#include "Constants.h"
#include "Encoding.h"
#include "FileMethods.h"
#include "Globals.h"

```

```

#include "SeriaMethods.h"
#include "StaffRender.h"

#ifndef NOMINMAX
#define NOMINMAX
#endif
#include <windows.h>
#include <commctrl.h>
#include <commdlg.h>
#include <dwmapi.h>
#include <iomanip>
#include <sstream>
#include <string>
#include <vector>

#pragma comment(lib, "Comctl32.lib")
#pragma comment(lib, "Comdlg32.lib")
#pragma comment(lib, "Dwmapi.lib")

namespace {

    constexpr COLORREF CLR_BG = RGB(0x1a, 0x1a, 0x1a);
    constexpr COLORREF CLR_BTN = RGB(0x2b, 0x2b, 0x2b);
    constexpr COLORREF CLR_BTN_HOVER = RGB(0x38, 0x38, 0x38);
    constexpr COLORREF CLR_BTN_BORDER = RGB(0x48, 0x48, 0x48);
    constexpr COLORREF CLR_TEXT = RGB(0xff, 0xff, 0xff);
    constexpr COLORREF CLR_STATUS_BAD = RGB(0xff, 0x55, 0x55);
    constexpr COLORREF CLR_STATUS_OK = RGB(0x55, 0xdd, 0x88);
    constexpr int MATRIX_CELL_CHARS = 9;
    constexpr int MAIN_CLIENT_W = 560;
    constexpr int MAIN_CLIENT_H = 620;

    HWND g_hwnd = nullptr;
    HFONT g_fontTitle = nullptr;
    HFONT g_fontBody = nullptr;
    HFONT g_fontBtn = nullptr;
    HFONT g_fontMono = nullptr;
    HBRUSH g_brBg = nullptr;

    std::string g_appDir;
    bool g_statusPositive = false;

    enum class Screen {
        MainEmpty,
        MainLoaded,
        Edit,
        NotesInput,
        Matrix,
    };

    Screen g_screen = Screen::MainEmpty;
    std::vector<HWND> g_tracked;

```

```

void build_ui();

Note g_tempNotes[SERIA_COUNT];
int g_notesSel = 0;
WNDPROC g_staffEditOldProc = nullptr;

enum CtrlId : int {
    IDC_HDR_TITLE = 101,
    IDC_HDR_STATUS = 102,
    IDC_HDR_HINT = 103,
    ID_ME_CREATE = 110,
    ID_ME_LOAD = 111,
    ID_ME_INFO = 112,
    ID_ME_EXIT = 113,
    ID_ML_EDIT = 120,
    ID_ML_SAVE = 121,
    ID_ML_NEW = 122,
    ID_ML_INFO = 123,
    ID_ML_EXIT = 124,
    ID_ED_SERIA = 130,
    ID_ED_INPUT = 131,
    ID_ED_ANALYSIS = 132,
    ID_ED_MATRIX = 133,
    ID_ED_BACK = 134,
    ID_NT_STAFF = 140,
    ID_NT_UP = 141,
    ID_NT_DOWN = 142,
    ID_NT_LEFT = 143,
    ID_NT_RIGHT = 144,
    ID_NT_DONE = 145,
    ID_MX_EDIT = 150,
    ID_MX_SAVE = 151,
    ID_MX_BACK = 152,
};

std::wstring utf8_to_wide(const std::string &u8) {
    if (u8.empty()) {
        return {};
    }
    const int n =
        MultiByteToWideChar(CP_UTF8, 0, u8.data(),
static_cast<int>(u8.size()), nullptr, 0);
    if (n <= 0) {
        return {};
    }
    std::wstring w(static_cast<size_t>(n), L'\0');
    MultiByteToWideChar(CP_UTF8, 0, u8.data(),
static_cast<int>(u8.size()), w.data(), n);
    return w;
}

std::string wide_to_utf8(const std::wstring &w) {
    if (w.empty()) {

```

```

        return {};
    }
    const int n = WideCharToMultiByte(CP_UTF8, 0, w.data(),
static_cast<int>(w.size()), nullptr, 0,
                                nullptr, nullptr);
    if (n <= 0) {
        return {};
    }
    std::string s(static_cast<size_t>(n), '\\0');
    WideCharToMultiByte(CP_UTF8, 0, w.data(),
static_cast<int>(w.size()), s.data(), n, nullptr, nullptr);
    return s;
}

void try_dark_title_bar(HWND hwnd) {
    BOOL dark = TRUE;
    using PFN = HRESULT(WINAPI *) (HWND, DWORD, LPCVOID,
DWORD);
    if (HMODULE dwm = LoadLibraryW(L"dwmapi.dll")) {
        if (auto p =
reinterpret_cast<PFN>(GetProcAddress(dwm,
"DwmSetWindowAttribute"))) {
            if (FAILED(p(hwnd, 20, &dark, sizeof(dark)))) {
                p(hwnd, 19, &dark, sizeof(dark));
            }
        }
        FreeLibrary(dwm);
    }
}

void track(HWND h) {
    if (h) {
        g_tracked.push_back(h);
    }
}

void destroy_tracked() {
    for (HWND h : g_tracked) {
        if (IsWindow(h)) {
            DestroyWindow(h);
        }
    }
    g_tracked.clear();
}

void invalidate_tracked() {
    if (g_hwnd) {
        InvalidateRect(g_hwnd, nullptr, TRUE);
    }
    for (HWND h : g_tracked) {
        if (IsWindow(h)) {
            InvalidateRect(h, nullptr, TRUE);
        }
    }
}

```

```

    }
}

void mono_font_metrics(int &charWidth, int &lineHeight) {
    charWidth = 8;
    lineHeight = 16;
    if (!g_hwnd || !g_fontMono) {
        return;
    }
    HDC hdc = GetDC(g_hwnd);
    if (!hdc) {
        return;
    }
    HGDIOBJ old = SelectObject(hdc, g_fontMono);
    TEXTMETRIC tm{};
    GetTextMetrics(hdc, &tm);
    SelectObject(hdc, old);
    ReleaseDC(g_hwnd, hdc);
    charWidth = tm.tmAveCharWidth;
    lineHeight = tm.tmHeight + tm.tmExternalLeading;
}

void mono_content_size(int cols, int lines, int &outW, int
&outH) {
    int charWidth = 0;
    int lineHeight = 0;
    mono_font_metrics(charWidth, lineHeight);
    outW = cols * charWidth + 24;
    outH = lines * lineHeight + 12;
}

void apply_font(HWND w, HFONT f) {
    if (w && f) {
        SendMessageW(w, WM_SETFONT,
reinterpret_cast<WPARAM>(f), TRUE);
    }
}

HWND mkLbl(HWND parent, int id, const wchar_t *text, DWORD
style = 0) {
    HWND h = CreateWindowExW(0, L"STATIC", text,
                                WS_CHILD | WS_VISIBLE |
SS_LEFT | SS_NOPREFIX | style, 0, 0, 100, 24,
                                parent,
reinterpret_cast<HMENU>(static_cast<INT_PTR>(id)),
                                GetModuleHandleW(nullptr),
nullptr);
    apply_font(h, g_fontBody);
    track(h);
    return h;
}

HWND mkBtnOd(HWND parent, int id, const wchar_t *text) {

```

```

        HWND h =
            CreateWindowExW(0, L"BUTTON", text,
                           WS_CHILD | WS_VISIBLE |
WS_TABSTOP | BS_OWNERDRAW, 0, 0, 100, 40, parent,
                           reinterpret_cast<HMENU>(static_cast<INT_PTR>(id)), GetModuleHandleW(nullptr), nullptr);
        track(h);
        return h;
    }

    HWND mkEdit(HWND parent, int id, DWORD style, HFONT font) {
        HWND h = CreateWindowExW(
            WS_EX_CLIENTEDGE, L"EDIT", L"",
            WS_CHILD | WS_VISIBLE | style, 0, 0, 100, 100,
parent,
            reinterpret_cast<HMENU>(static_cast<INT_PTR>(id)),
GetModuleHandleW(nullptr), nullptr);
        apply_font(h, font);
        track(h);
        return h;
    }

    HWND mkList(HWND parent, int id) {
        HWND h =
            CreateWindowExW(WS_EX_CLIENTEDGE, L"LISTBOX", L"",
                           WS_CHILD | WS_VISIBLE |
LBS_NOTIFY | WS_VSCROLL | LBS_NOINTEGRALHEIGHT, 0, 0,
                           100, 100, parent,
            reinterpret_cast<HMENU>(static_cast<INT_PTR>(id)),
                           GetModuleHandleW(nullptr),
nullptr);
        apply_font(h, g_fontBody);
        track(h);
        return h;
    }

    void layout_main_headers(int cw, int margin, int &outBtnTop)
    {
        const int titleY = margin;
        const int statusY = titleY + 40;
        const int hintY = statusY + 26;
        outBtnTop = hintY + 36;
        MoveWindow(GetDlgItem(g_hwnd, IDC_HDR_TITLE), margin,
titleY, cw - 2 * margin, 36, TRUE);
        MoveWindow(GetDlgItem(g_hwnd, IDC_HDR_STATUS), margin,
statusY, cw - 2 * margin, 24, TRUE);
        MoveWindow(GetDlgItem(g_hwnd, IDC_HDR_HINT), margin,
hintY, cw - 2 * margin, 44, TRUE);
    }

    void matrix_content_size(int &outW, int &outH);
    void staff_content_size(int &outW, int &outH);
    void fit_window_client(int clientW, int clientH);

```

```

void fit_window_for_screen();
void fit_window_to_main();
std::wstring matrix_text_wide();

void layout_screen(int cw, int ch) {
    const int margin = 32;
    const int btnH = 42;
    const int gap = 14;

    if (g_screen == Screen::MainEmpty || g_screen ==
Screen::MainLoaded) {
        int btnTop = 0;
        layout_main_headers(cw, margin, btnTop);
        const int btnW = cw - 2 * margin;
        const int btnX = margin;
        int y = btnTop;
        if (g_screen == Screen::MainEmpty) {
            const int ids[] = {ID_ME_CREATE, ID_ME_LOAD,
ID_ME_INFO, ID_ME_EXIT};
            for (int id : ids) {
                HWND b = GetDlgItem(g_hwnd, id);
                if (b) {
                    MoveWindow(b, btnX, y, btnW, btnH,
TRUE);
                    y += btnH + gap;
                }
            }
        } else {
            const int ids[] = {ID_ML_EDIT, ID_ML_SAVE,
ID_ML_NEW, ID_ML_INFO, ID_ML_EXIT};
            for (int id : ids) {
                HWND b = GetDlgItem(g_hwnd, id);
                if (b) {
                    MoveWindow(b, btnX, y, btnW, btnH,
TRUE);
                    y += btnH + gap;
                }
            }
        }
        return;
    }

    if (g_screen == Screen::Edit) {
        int btnTop = 0;
        layout_main_headers(cw, margin, btnTop);
        HWND ed = GetDlgItem(g_hwnd, ID_ED_SERIA);
        const int edTop = btnTop;
        int staffW = 0;
        int staffH = 0;
        staff_content_size(staffW, staffH);
        if (ed) {
            MoveWindow(ed, margin, edTop, staffW, staffH,
TRUE);

```

```

    }
    int y = edTop + staffH + gap;
    const int btnW = cw - 2 * margin;
    for (int id : {ID_ED_INPUT, ID_ED_ANALYSIS,
ID_ED_MATRIX, ID_ED_BACK}) {
        HWND b = GetDlgItem(g_hwnd, id);
        if (b) {
            MoveWindow(b, margin, y, btnW, btnH,
TRUE);
            y += btnH + gap;
        }
    }
    return;
}

if (g_screen == Screen::NotesInput) {
    if (HWND hint = GetDlgItem(g_hwnd, IDC_HDR_HINT)) {
        ShowWindow(hint, SW_HIDE);
    }
    const int titleH = 32;
    const int statusH = 28;
    MoveWindow(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
margin, margin, cw - 2 * margin, titleH, TRUE);
    MoveWindow(GetDlgItem(g_hwnd, IDC_HDR_STATUS),
margin, margin + titleH + 4, cw - 2 * margin,
statusH, TRUE);
    HWND staffEd = GetDlgItem(g_hwnd, ID_NT_STAFF);
    const int staffTop = margin + titleH + 4 + statusH +
gap;

    int staffW = 0;
    int staffH = 0;
    staff_content_size(staffW, staffH);
    if (staffEd) {
        MoveWindow(staffEd, margin, staffTop, staffW,
staffH, TRUE);
    }
    const int rowY = staffTop + staffH + gap;
    const int bw = (cw - 2 * margin - 3 * gap) / 4;
    int x = margin;
    MoveWindow(GetDlgItem(g_hwnd, ID_NT_UP), x, rowY,
bw, btnH, TRUE);
    x += bw + gap;
    MoveWindow(GetDlgItem(g_hwnd, ID_NT_DOWN), x, rowY,
bw, btnH, TRUE);
    x += bw + gap;
    MoveWindow(GetDlgItem(g_hwnd, ID_NT_LEFT), x, rowY,
bw, btnH, TRUE);
    x += bw + gap;
    MoveWindow(GetDlgItem(g_hwnd, ID_NT_RIGHT), x, rowY,
bw, btnH, TRUE);
    MoveWindow(GetDlgItem(g_hwnd, ID_NT_DONE), margin,
rowY + btnH + gap, cw - 2 * margin, btnH,
TRUE);
}

```

```

        return;
    }

    if (g_screen == Screen::Matrix) {
        if (HWND hint = GetDlgItem(g_hwnd, IDC_HDR_HINT)) {
            ShowWindow(hint, SW_HIDE);
        }
        const int titleH = 36;
        const int statusH = 28;
        MoveWindow(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
margin, margin, cw - 2 * margin, titleH, TRUE);
        MoveWindow(GetDlgItem(g_hwnd, IDC_HDR_STATUS),
margin, margin + titleH + 4, cw - 2 * margin,
            statusH, TRUE);
        HWND ed = GetDlgItem(g_hwnd, ID_MX_EDIT);
        const int edTop = margin + titleH + 4 + statusH +
gap;

        int matrixW = 0;
        int matrixH = 0;
        matrix_content_size(matrixW, matrixH);
        const int edX = margin + ((cw - 2 * margin -
matrixW) > 0 ? (cw - 2 * margin - matrixW) / 2 : 0);
        if (ed) {
            MoveWindow(ed, edX, edTop, matrixW, matrixH,
TRUE);
        }
        const int y = edTop + matrixH + gap;
        const int btnW = cw - 2 * margin;
        MoveWindow(GetDlgItem(g_hwnd, ID_MX_SAVE), margin,
y, btnW, btnH, TRUE);
        MoveWindow(GetDlgItem(g_hwnd, ID_MX_BACK), margin, y
+ btnH + gap, btnW, btnH, TRUE);
    }
}

void msg_box_u8(const wchar_t *title, const std::string &u8)
{
    MessageBoxW(g_hwnd, utf8_to_wide(u8).c_str(), title,
MB_OK | MB_ICONINFORMATION);
}

bool confirm_warn(const wchar_t *body) {
    return MessageBoxW(g_hwnd, body, L"SeriaWizard", MB_YESNO
| MB_ICONWARNING) == IDYES;
}

void split_path_utf8(const std::string &full, std::string
&dirOut, std::string &fileOut) {
    const size_t slash = full.find_last_of("\\\\/");
    if (slash == std::string::npos) {
        dirOut = g_appDir;
        fileOut = full;
    } else {

```

```

        dirOut = full.substr(0, slash);
        fileOut = full.substr(slash + 1);
    }
}

bool dialog_open(std::string &fullPathUtf8) {
    wchar_t buf[MAX_PATH] = {0};
    OPENFILENAMEW ofn{};
    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = g_hwnd;
    ofn.lpstrFile = buf;
    ofn.nMaxFile = MAX_PATH;
    ofn.lpstrFilter = L"Текст (*.txt)\0*.txt\0Все
файлы\0*.*\0\0";
    ofn.Flags = OFN_FILEMUSTEXIST | OFN_PATHMUSTEXIST;
    std::wstring initial = utf8_to_wide(g_appDir);
    ofn.lpstrInitialDir = initial.c_str();
    if (!GetOpenFileNameW(&ofn)) {
        return false;
    }
    fullPathUtf8 = wide_to_utf8(buf);
    return true;
}

bool dialog_save(std::string &fullPathUtf8) {
    wchar_t buf[MAX_PATH] = {0};
    OPENFILENAMEW ofn{};
    ofn.lStructSize = sizeof(ofn);
    ofn.hwndOwner = g_hwnd;
    ofn.lpstrFile = buf;
    ofn.nMaxFile = MAX_PATH;
    ofn.lpstrFilter = L"Текст (*.txt)\0*.txt\0Все
файлы\0*.*\0\0";
    ofn.Flags = OFN_OVERWRITEPROMPT | OFN_PATHMUSTEXIST;
    std::wstring initial = utf8_to_wide(g_appDir);
    ofn.lpstrInitialDir = initial.c_str();
    ofn.lpstrDefExt = L"txt";
    if (!GetSaveFileNameW(&ofn)) {
        return false;
    }
    fullPathUtf8 = wide_to_utf8(buf);
    return true;
}

void refresh_staff_edit(HWND ed, const Note
notes[SERIA_COUNT], int selectedIndex) {
    if (!ed) {
        return;
    }
    const std::string staff =
StaffRender::renderStaffUtf8(notes, selectedIndex);
    std::wstring text = utf8_to_wide(staff);
    std::wstring crlf;

```

```

        crlf.reserve(text.size() + 32);
        for (wchar_t ch : text) {
            if (ch == L'\n') {
                crlf += L"\r\n";
            } else {
                crlf += ch;
            }
        }
        SetWindowTextW(ed, crlf.c_str());
        InvalidateRect(ed, nullptr, TRUE);
    }

    void handle_notes_input(int id) {
        switch (id) {
            case ID_NT_UP:
                StaffRender::stepPitchUp(g_tempNotes[g_notesSel]);
                break;
            case ID_NT_DOWN:
                StaffRender::stepPitchDown(g_tempNotes[g_notesSel]);
                break;
            case ID_NT_LEFT:
                g_notesSel = (g_notesSel - 1 + SERIA_COUNT) %
SERIA_COUNT;
                break;
            case ID_NT_RIGHT:
                g_notesSel = (g_notesSel + 1) % SERIA_COUNT;
                break;
            case ID_NT_DONE:
                for (int i = 0; i < SERIA_COUNT; ++i) {
                    thisSeria[i] = g_tempNotes[i];
                }
                SeriaMethods::collectIntervals(thisSeria,
thisIntervals);
                isSeriaInitialized = true;
                analysedSeriaSwitch = 0;
                g_screen = Screen::Edit;
                build_ui();
                return;
            default:
                return;
        }
        refresh_staff_edit(GetDlgItem(g_hwnd, ID_NT_STAFF),
g_tempNotes, g_notesSel);
    }

    LRESULT CALLBACK StaffInputEditProc(HWND hwnd, UINT msg,
WPARAM wParam, LPARAM lParam) {
        if (msg == WM_KEYDOWN && g_screen == Screen::NotesInput)
        {
            switch (wParam) {
                case VK_LEFT:
                    handle_notes_input(ID_NT_LEFT);
                    return 0;
            }
        }
    }

```

```

        case VK_RIGHT:
            handle_notes_input(ID_NT_RIGHT);
            return 0;
        case VK_UP:
            handle_notes_input(ID_NT_UP);
            return 0;
        case VK_DOWN:
            handle_notes_input(ID_NT_DOWN);
            return 0;
        case VK_RETURN:
            handle_notes_input(ID_NT_DONE);
            return 0;
        default:
            break;
    }
}

return CallWindowProcW(g_staffEditOldProc, hwnd, msg,
wParam, lParam);
}

void matrix_content_size(int &outW, int &outH) {
    mono_content_size(MATRIX_CELL_CHARS * SERIA_COUNT,
SERIA_COUNT, outW, outH);
}

void staff_content_size(int &outW, int &outH) {
    mono_content_size(StaffRender::STAFF_COLS,
StaffRender::STAFF_ROWS + 3, outW, outH);
}

std::wstring matrix_text_wide() {
    std::wstring out;
    for (int r = 0; r < SERIA_COUNT; ++r) {
        for (int c = 0; c < SERIA_COUNT; ++c) {
            const Note &cell = thisMatrix[r * SERIA_COUNT +
c];

            std::ostringstream oss;
            oss << std::setw(MATRIX_CELL_CHARS) <<
std::left << SeriaMethods::matrixCellToString(cell);
            out += utf8_to_wide(oss.str());
        }
        if (r + 1 < SERIA_COUNT) {
            out += L"\r\n";
        }
    }
    return out;
}

void fit_window_client(int clientW, int clientH) {
    if (!g_hwnd) {
        return;
    }
    RECT wr{};

```

```

        GetWindowRect(g_hwnd, &wr);
        const DWORD style =
static_cast<DWORD>(GetWindowLongW(g_hwnd, GWL_STYLE));
        const DWORD exStyle =
static_cast<DWORD>(GetWindowLongW(g_hwnd, GWL_EXSTYLE));
        RECT frame = {0, 0, clientW, clientH};
        AdjustWindowRectEx(&frame, style, FALSE, exStyle);
        const int winW = frame.right - frame.left;
        const int winH = frame.bottom - frame.top;
        SetWindowPos(g_hwnd, nullptr, wr.left, wr.top, winW,
winH, SWP_NOZORDER | SWP_NOMOVE);
    }

    void fit_window_for_screen() {
        const int margin = 32;
        const int btnH = 42;
        const int gap = 14;

        if (g_screen == Screen::Matrix) {
            int contentW = 0;
            int contentH = 0;
            matrix_content_size(contentW, contentH);
            const int headerH = 82;
            fit_window_client(margin * 2 + contentW,
                             margin + headerH + contentH +
gap + 2 * btnH + gap + margin);
            return;
        }

        if (g_screen == Screen::Edit) {
            int contentW = 0;
            int contentH = 0;
            staff_content_size(contentW, contentH);
            int btnTop = 0;
            layout_main_headers(contentW + 2 * margin, margin,
btnTop);

            const int clientW = margin * 2 + contentW;
            const int clientH = btnTop + contentH + gap + 4 *
(btnH + gap) + margin;
            fit_window_client(clientW, clientH);
            return;
        }

        if (g_screen == Screen::NotesInput) {
            int contentW = 0;
            int contentH = 0;
            staff_content_size(contentW, contentH);
            const int headerH = 78;
            const int clientW = margin * 2 + contentW;
            const int clientH = margin + headerH + contentH +
gap + 2 * btnH + gap + margin;
            fit_window_client(clientW, clientH);
        }
    }

```

```

}

void fit_window_to_main() {
    fit_window_client(MAIN_CLIENT_W, MAIN_CLIENT_H);
}

void go_edit_after_create() {
    analysedSerialSwitch = 0;
    g_screen = Screen::Edit;
}

void leave_edit_back_to_main() {
    if (isSerialInitialized) {
        g_screen = Screen::MainLoaded;
        g_statusPositive = true;
    } else {
        g_screen = Screen::MainEmpty;
        g_statusPositive = false;
    }
}

void on_command(int id) {
    switch (id) {
        case ID_ME_CREATE:
            resetSerialState();
            go_edit_after_create();
            build_ui();
            break;
        case ID_ME_LOAD: {
            std::string path;
            if (!dialog_open(path)) {
                break;
            }
            std::string dir;
            std::string fname;
            split_path_utf8(path, dir, fname);
            if (!loadSerialFromFile(dir, fname, thisSerial,
SERIAL_COUNT)) {
                msg_box_u8(L"О Ш И Б К А!", u8"Не удалось
открыть файл или прочесть данные.");
            } else {
                SerialMethods::collectIntervals(thisSerial,
thisIntervals);

                isSerialInitialized = true;
                g_statusPositive = true;
                g_screen = Screen::MainLoaded;
                build_ui();
            }
            break;
        }
        case ID_ME_INFO:
            msg_box_u8(L"Информация", INFORMATION_MESSAGE);
            break;
    }
}

```

```

        case ID_ME_EXIT:
            if (confirm_warn(L"Несохранённые данные будут
утеряны. Выйти?")) {
                PostQuitMessage(0);
            }
            break;

        case ID_ML_EDIT:
            g_screen = Screen::Edit;
            build_ui();
            break;
        case ID_ML_SAVE: {
            std::string path;
            if (!dialog_save(path)) {
                break;
            }
            std::string dir;
            std::string fname;
            split_path_utf8(path, dir, fname);
            if (!saveSeriaToFile(dir, fname, thisSeria,
SERIA_COUNT)) {
                msg_box_u8(L"О Ш И Б К А!", u8"Не удалось
сохранить файл.");
            }
            break;
        }
        case ID_ML_NEW:
            if (confirm_warn(L"Несохранённые данные будут
утеряны. Продолжить?")) {
                resetSeriaState();
                alalysResult = 0;
                g_statusPositive = false;
                g_screen = Screen::MainEmpty;
                build_ui();
            }
            break;
        case ID_ML_INFO:
            msg_box_u8(L"Б Е Р Е Г И Т Е С Ь!",
INFORMATION_MESSAGE);
            break;
        case ID_ML_EXIT:
            if (confirm_warn(L"Несохранённые данные будут
утеряны. Выйти?")) {
                PostQuitMessage(0);
            }
            break;

        case ID_ED_INPUT:
            analysedSeriaSwitch = 0;
            for (int i = 0; i < SERIA_COUNT; ++i) {
                g_tempNotes[i] = thisSeria[i];
            }
            g_notesSel = 0;

```

```

        g_screen = Screen::NotesInput;
        build_ui();
        break;
    case ID_ED_ANALYSIS:
        SerialMethods::runAnalysis(analysResultMessage,
analysResult);
        msg_box_u8(L"Анализ", analysResultMessage);
        break;
    case ID_ED_MATRIX:
        if (analysResult <= 0) {
            msg_box_u8(L"О Ш И Б К А!",
                u8"Невозможно построить матрицу: либо
анализ провален, либо не проведён.");
        } else {
            SerialMethods::buildMatrix(thisSeria,
thisMatrix);

            g_screen = Screen::Matrix;
            build_ui();
        }
        break;
    case ID_ED_BACK:
        leave_edit_back_to_main();
        build_ui();
        break;

    case ID_NT_UP:
    case ID_NT_DOWN:
    case ID_NT_LEFT:
    case ID_NT_RIGHT:
    case ID_NT_DONE:
        handle_notes_input(id);
        break;

    case ID_MX_SAVE: {
        std::string path;
        if (!dialog_save(path)) {
            break;
        }
        std::string dir;
        std::string fname;
        split_path_utf8(path, dir, fname);
        if (!saveTextToFile(dir, fname,
wide_to_utf8(matrix_text_wide())) {
            msg_box_u8(L"О Ш И Б К А!", u8"Не удалось
сохранить файл.");
        }
        break;
    }
    case ID_MX_BACK:
        g_screen = Screen::Edit;
        build_ui();
        break;
    default:

```

```

        break;
    }
}

void set_header_main_empty() {
    SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
L"SeriaWizard");
    SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_STATUS),
L"Данные не загружены");
    SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_HINT),
L"Создайте новую серию или откройте .txt с серией");
    apply_font(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
g_fontTitle);
}

void set_header_main_loaded() {
    SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
L"SeriaWizard");
    SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_STATUS),
L"Данные загружены");
    SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_HINT),
L"Редактируйте серию, сохраните в
файл или создайте новую.");
    apply_font(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
g_fontTitle);
}

void build_ui() {
    destroy_tracked();

    RECT rc{};
    GetClientRect(g_hwnd, &rc);
    int cw = rc.right - rc.left;
    int ch = rc.bottom - rc.top;

    mkLbl(g_hwnd, IDC_HDR_TITLE, L"", 0);
    mkLbl(g_hwnd, IDC_HDR_STATUS, L"", 0);
    mkLbl(g_hwnd, IDC_HDR_HINT, L"", 0);

    switch (g_screen) {
case Screen::MainEmpty:
    g_statusPositive = false;
    set_header_main_empty();
    mkBtnOd(g_hwnd, ID_ME_CREATE, L"Создать серию");
    mkBtnOd(g_hwnd, ID_ME_LOAD, L"Загрузить серию");
    mkBtnOd(g_hwnd, ID_ME_INFO, L"Информация");
    mkBtnOd(g_hwnd, ID_ME_EXIT, L"Выход");
    break;

case Screen::MainLoaded:
    g_statusPositive = true;
    set_header_main_loaded();
    mkBtnOd(g_hwnd, ID_ML_EDIT, L"Редактировать серию");

```

```

        mkBtnOd(g_hwnd, ID_ML_SAVE, L"Сохранить серию");
        mkBtnOd(g_hwnd, ID_ML_NEW, L"Новая серия");
        mkBtnOd(g_hwnd, ID_ML_INFO, L"Информация");
        mkBtnOd(g_hwnd, ID_ML_EXIT, L"Выход");
        break;

    case Screen::Edit:
        g_statusPositive = true;
        SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
L"SeriaWizard");
        SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_STATUS),
L"Редактирование серии");
        SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_HINT),
L"");
        apply_font(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
g_fontTitle);
        {
            HWND ed = mkEdit(g_hwnd, ID_ED_SERIA,
ES_MULTILINE | ES_READONLY, g_fontMono);
            refresh_staff_edit(ed, thisSeria, -1);
        }
        mkBtnOd(g_hwnd, ID_ED_INPUT, L"Ввод тонов");
        mkBtnOd(g_hwnd, ID_ED_ANALYSIS, L"Анализ серии");
        mkBtnOd(g_hwnd, ID_ED_MATRIX, L"Генерация матрицы");
        mkBtnOd(g_hwnd, ID_ED_BACK, L"Назад");
        break;

    case Screen::NotesInput:
        SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
L"SeriaWizard");
        SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_STATUS),
L"Ввод тонов");
        SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_HINT),
L"");
        apply_font(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
g_fontTitle);
        {
            HWND ed = mkEdit(g_hwnd, ID_NT_STAFF,
ES_MULTILINE | ES_READONLY, g_fontMono);
            refresh_staff_edit(ed, g_tempNotes,
g_notesSel);
            g_staffEditOldProc =
reinterpret_cast<WNDPROC>(SetWin
dowLongPtrW(ed, GWLP_WNDPROC,
reinterpret_cast<LONG_PTR>(Staff
InputEditProc)));
            SetFocus(ed);
        }
        mkBtnOd(g_hwnd, ID_NT_UP, L"↑ Повысить тон");
        mkBtnOd(g_hwnd, ID_NT_DOWN, L"↓ Понизить тон");
        mkBtnOd(g_hwnd, ID_NT_LEFT, L"← Влево");
        mkBtnOd(g_hwnd, ID_NT_RIGHT, L"Вправо →");
        mkBtnOd(g_hwnd, ID_NT_DONE, L"Готово");

```

```

        break;

        case Screen::Matrix:
            SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
L"SeriaWizard");
            SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_STATUS),
L"Матрица серийных форм");
            SetWindowTextW(GetDlgItem(g_hwnd, IDC_HDR_HINT),
L"");
            apply_font(GetDlgItem(g_hwnd, IDC_HDR_TITLE),
g_fontTitle);
            {
                HWND ed = mkEdit(g_hwnd, ID_MX_EDIT,
ES_MULTILINE | ES_READONLY, g_fontMono);
                SetWindowTextW(ed, matrix_text_wide().c_str());
                SendMessageW(ed, WM_SETFONT,
reinterpret_cast<WPARAM>(g_fontMono), TRUE);
                InvalidateRect(ed, nullptr, TRUE);
            }
            mkBtnOd(g_hwnd, ID_MX_SAVE, L"Сохранить в файл");
            mkBtnOd(g_hwnd, ID_MX_BACK, L"Назад");
            break;
        }

        if (g_screen == Screen::Matrix || g_screen ==
Screen::Edit || g_screen == Screen::NotesInput) {
            fit_window_for_screen();
            GetClientRect(g_hwnd, &rc);
            cw = rc.right - rc.left;
            ch = rc.bottom - rc.top;
        } else if (g_screen == Screen::MainEmpty || g_screen ==
Screen::MainLoaded) {
            fit_window_to_main();
            GetClientRect(g_hwnd, &rc);
            cw = rc.right - rc.left;
            ch = rc.bottom - rc.top;
        }

        layout_screen(cw, ch);
        invalidate_tracked();
    }

    void draw_owner_button(const DRAWITEMSTRUCT *dis) {
        HDC hdc = dis->hDC;
        RECT rc = dis->rcItem;
        InflateRect(&rc, -2, -2);

        const bool pressed = (dis->itemState & ODS_SELECTED) !=
0;

        const COLORREF fill = pressed ? CLR_BTN_HOVER : CLR_BTN;

        HBRUSH br = CreateSolidBrush(fill);
        HPEN pen = CreatePen(PS_SOLID, 1, CLR_BTN_BORDER);

```

```

        HGDIOBJ oldBr = SelectObject(hdc, br);
        HGDIOBJ oldPen = SelectObject(hdc, pen);
        RoundRect(hdc, rc.left, rc.top, rc.right, rc.bottom, 12,
12);
        SelectObject(hdc, oldPen);
        SelectObject(hdc, oldBr);
        DeleteObject(pen);
        DeleteObject(br);

        wchar_t text[256]{};
        GetWindowTextW(dis->hwndItem, text, 255);
        SetBkMode(hdc, TRANSPARENT);
        SetTextColor(hdc, CLR_TEXT);
        SelectObject(hdc, g_fontBtn);
        DrawTextW(hdc, text, -1, const_cast<RECT *>(&dis-
>rcItem), DT_CENTER | DT_VCENTER | DT_SINGLELINE);
    }

```

```

        LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam,
LPARAM lParam) {
            switch (msg) {
            case WM_CREATE:
                g_hwnd = hwnd;
                g_brBg = CreateSolidBrush(CLR_BG);
                g_fontTitle =
                    CreateFontW(-28, 0, 0, 0, FW_BOLD, FALSE,
FALSE, FALSE, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, L"Segoe UI");
                g_fontBody =
                    CreateFontW(-15, 0, 0, 0, FW_NORMAL, FALSE,
FALSE, FALSE, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, L"Segoe UI");
                g_fontBtn =
                    CreateFontW(-16, 0, 0, 0, FW_NORMAL, FALSE,
FALSE, FALSE, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY, DEFAULT_PITCH | FF_DONTCARE, L"Segoe UI");
                g_fontMono =
                    CreateFontW(-15, 0, 0, 0, FW_NORMAL, FALSE,
FALSE, FALSE, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS,
                    CLIP_DEFAULT_PRECIS,
DEFAULT_QUALITY, FIXED_PITCH | FF_MODERN, L"Consolas");
                try_dark_title_bar(hwnd);
                g_appDir = applicationDirectory();
                resetSeriaState();
                g_screen = Screen::MainEmpty;
                build_ui();
                return 0;

            case WM_DRAWITEM: {

```

```

auto *dis = reinterpret_cast<DRAWITEMSTRUCT
*>(lParam);

    if (dis->CtlType == ODT_BUTTON) {
        draw_owner_button(dis);
        return TRUE;
    }
    break;
}

case WM_ERASEBKGND: {
    HDC hdc = reinterpret_cast<HDC>(wParam);
    RECT rc{};
    GetClientRect(hwnd, &rc);
    FillRect(hdc, &rc, g_brBg);
    return 1;
}

case WM_PAINT: {
    PAINTSTRUCT ps{};
    HDC hdc = BeginPaint(hwnd, &ps);
    RECT cr{};
    GetClientRect(hwnd, &cr);
    FillRect(hdc, &cr, g_brBg);
    EndPaint(hwnd, &ps);
    return 0;
}

case WM_CTLCOLORSTATIC: {
    HDC hdc = reinterpret_cast<HDC>(wParam);
    HWND st = reinterpret_cast<HWND>(lParam);
    SetBkMode(hdc, TRANSPARENT);
    const int id = GetDlgCtrlID(st);
    if (id == IDC_HDR_STATUS &&
        (g_screen == Screen::MainEmpty || g_screen ==
Screen::MainLoaded)) {
        SetTextColor(hdc, g_statusPositive ?
CLR_STATUS_OK : CLR_STATUS_BAD);
    } else {
        SetTextColor(hdc, CLR_TEXT);
    }
    return reinterpret_cast<INT_PTR>(g_brBg);
}

case WM_CTLCOLOREDIT: {
    HDC hdc = reinterpret_cast<HDC>(wParam);
    SetBkColor(hdc, RGB(0x22, 0x22, 0x22));
    SetTextColor(hdc, CLR_TEXT);
    static HBRUSH brEd = CreateSolidBrush(RGB(0x22,
0x22, 0x22));
    return reinterpret_cast<INT_PTR>(brEd);
}

case WM_CTLCOLORLISTBOX: {

```

```

        HDC hdc = reinterpret_cast<HDC>(wParam);
        SetBkColor(hdc, RGB(0x22, 0x22, 0x22));
        SetTextColor(hdc, CLR_TEXT);
        static HBRUSH brLb = CreateSolidBrush(RGB(0x22,
0x22, 0x22));
        return reinterpret_cast<INT_PTR>(brLb);
    }

    case WM_KEYDOWN:
        if (g_screen == Screen::NotesInput) {
            switch (wParam) {
                case VK_LEFT:
                    handle_notes_input(ID_NT_LEFT);
                    return 0;
                case VK_RIGHT:
                    handle_notes_input(ID_NT_RIGHT);
                    return 0;
                case VK_UP:
                    handle_notes_input(ID_NT_UP);
                    return 0;
                case VK_DOWN:
                    handle_notes_input(ID_NT_DOWN);
                    return 0;
                case VK_RETURN:
                    handle_notes_input(ID_NT_DONE);
                    return 0;
                default:
                    break;
            }
        }
        break;

    case WM_COMMAND:
        if (HIWORD(wParam) == BN_CLICKED) {
            on_command(static_cast<int>(LOWORD(wParam)));
        }
        return 0;

    case WM_SIZE:
        if (wParam != SIZE_MINIMIZED && g_hwnd == hwnd &&
g_brBg) {
            layout_screen(LOWORD(lParam), HIWORD(lParam));
            invalidate_tracked();
        }
        return 0;

    case WM_GETMINMAXINFO: {
        auto *mmi = reinterpret_cast<MINMAXINFO *>(lParam);
        mmi->ptMinTrackSize.x = 480;
        mmi->ptMinTrackSize.y = 440;
        return 0;
    }
}

```

```

        case WM_DESTROY:
            destroy_tracked();
            if (g_fontTitle) {
                DeleteObject(g_fontTitle);
                g_fontTitle = nullptr;
            }
            if (g_fontBody) {
                DeleteObject(g_fontBody);
                g_fontBody = nullptr;
            }
            if (g_fontBtn) {
                DeleteObject(g_fontBtn);
                g_fontBtn = nullptr;
            }
            if (g_fontMono) {
                DeleteObject(g_fontMono);
                g_fontMono = nullptr;
            }
            if (g_brBg) {
                DeleteObject(g_brBg);
                g_brBg = nullptr;
            }
            PostQuitMessage(0);
            return 0;

        default:
            break;
    }
    return DefWindowProcW(hwnd, msg, wParam, lParam);
}

int WINAPI WinMain(HINSTANCE hInst, HINSTANCE, LPSTR, int
nCmdShow) {
    INITCOMMONCONTROLSEX icc{};
    icc.dwSize = sizeof(icc);
    icc.dwICC = ICC_STANDARD_CLASSES;
    InitCommonControlsEx(&icc);

    const wchar_t *cls = L"SeriaWizardDarkWnd";
    WNDCLASSW wc{};
    wc.lpfnWndProc = WndProc;
    wc.hInstance = hInst;
    wc.lpszClassName = cls;
    wc.hCursor = LoadCursor(nullptr, IDC_ARROW);
    wc.hbrBackground = nullptr;
    RegisterClassW(&wc);

    RECT rc{0, 0, MAIN_CLIENT_W, MAIN_CLIENT_H};
    const DWORD frameStyle = WS_OVERLAPPEDWINDOW |
WS_CLIPCHILDREN;
    AdjustWindowRect(&rc, frameStyle, FALSE);
    const int ww = rc.right - rc.left;

```

```

    const int wh = rc.bottom - rc.top;

    HWND hwnd = CreateWindowExW(WS_EX_APPWINDOW, cls,
L"SeriaWizard", frameStyle, CW_USEDEFAULT,
                                CW_USEDEFAULT, ww, wh,
nullptr, nullptr, hInst, nullptr);
    ShowWindow(hwnd, nCmdShow);

    MSG msg{};
    while (GetMessageW(&msg, nullptr, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessageW(&msg);
    }
    return static_cast<int>(msg.wParam);
}

```

Листинг Б.3 – Код «Note.h»

```

#pragma once

#include <string>

class Note {
private:
    char letter_;
    int alteration_;
    int octave_;

    void moveOctaveBoundary(bool up);
    char stepUp() const;
    char stepDown() const;

public:
    Note();
    Note(char letter, int alteration, int octave);

    char getLetter() const { return letter_; }
    void setLetter(char L);
    int getAlteration() const { return alteration_; }
    void setAlteration(int A);
    int getOctave() const { return octave_; }
    void setOctave(int O);

    void enharmonicChange();
    void upAlteration();
    void downAlteration();

    std::string toString(bool omitOctave = true) const;
    std::string toStringGerman(bool omitOctave = true) const;
    int getMidi() const;

    bool operator>(const Note &o) const;
    bool operator>=(const Note &o) const;
    bool operator<(const Note &o) const;

```

```

    bool operator<=(const Note &o) const;
    bool operator==(const Note &o) const;
    bool operator!=(const Note &o) const;
    bool equalsWithOctave(const Note &o) const;
};

```

Листинг Б.4 – Код «Note.cpp»

```

#include "Note.h"
#include "Constants.h"
#include "NoteMethods.h"
#include <cctype>
#include <sstream>

Note::Note() : letter_('B'), alteration_(0), octave_(4) {}

Note::Note(char letter, int alteration, int octave)
    : letter_(letter), alteration_(alteration), octave_(octave)
{}

void Note::setLetter(char L) {
    if (NOTE_LETTERS.count(L)) {
        letter_ = L;
    }
}

void Note::setAlteration(int A) {
    if (A >= -2 && A <= 2) {
        alteration_ = A;
    }
}

void Note::setOctave(int O) {
    if (O >= 0 && O <= 7) {
        octave_ = O;
    }
}

void Note::moveOctaveBoundary(bool up) {
    if (up) {
        setOctave(getOctave() + 1);
        setLetter('C');
    } else {
        setOctave(getOctave() - 1);
        setLetter('B');
    }
}

char Note::stepUp() const {
    return fromStepToLetter(fromLetterToStep(letter_) + 1);
}

char Note::stepDown() const {
    return fromStepToLetter(fromLetterToStep(letter_) - 1);
}

```

```

}

void Note::enharmonicChange() {
    const bool upEnharmonic =
        (letter_ == 'C' && alteration_ == 2) || (letter_ == 'D'
&& alteration_ == 2) ||
        (letter_ == 'E' && alteration_ == 1) || (letter_ == 'F'
&& alteration_ == 2) ||
        (letter_ == 'G' && alteration_ == 2) || (letter_ == 'A'
&& alteration_ == 2) ||
        (letter_ == 'B' && alteration_ == 1) || (letter_ == 'B'
&& alteration_ == 2);

    if (upEnharmonic) {
        const bool bx = (letter_ == 'B' && alteration_ == 2);
        int s = fromLetterToStep(letter_) + 1;
        if (s > 7) {
            moveOctaveBoundary(true);
            s = 1;
        }
        setLetter(fromStepToLetter(s));
        setAlteration(bx ? 1 : 0);
        return;
    }

    const bool downEnharmonic =
        (letter_ == 'D' && alteration_ == -2) || (letter_ == 'E'
&& alteration_ == -2) ||
        (letter_ == 'F' && alteration_ == -1) || (letter_ == 'G'
&& alteration_ == -2) ||
        (letter_ == 'A' && alteration_ == -2) || (letter_ == 'B'
&& alteration_ == -2) ||
        (letter_ == 'C' && alteration_ == -1) || (letter_ == 'C'
&& alteration_ == -2);

    if (downEnharmonic) {
        const bool cbb = (letter_ == 'C' && alteration_ == -2);
        int s = fromLetterToStep(letter_) - 1;
        if (s < 1) {
            moveOctaveBoundary(false);
            s = 7;
        }
        setLetter(fromStepToLetter(s));
        setAlteration(cbb ? -1 : 0);
    }
}

void Note::upAlteration() {
    if (getAlteration() == 2) {
        const char oldLetter = letter_;
        int step = fromLetterToStep(letter_) + 1;
        if (step > 7) {
            moveOctaveBoundary(true);

```

```

        step = 1;
    }
    setLetter(fromStepToLetter(step));
    if (oldLetter == 'E' || oldLetter == 'B') {
        setAlteration(2);
    } else {
        setAlteration(1);
    }
} else if (getAlteration() < 2) {
    setAlteration(getAlteration() + 1);
}
}

void Note::downAlteration() {
    if (getAlteration() == -2) {
        const char oldLetter = letter_;
        int step = fromLetterToStep(letter_) - 1;
        if (step < 1) {
            moveOctaveBoundary(false);
            step = 7;
        }
        setLetter(fromStepToLetter(step));
        if (oldLetter == 'F' || oldLetter == 'C') {
            setAlteration(-2);
        } else {
            setAlteration(-1);
        }
    } else if (getAlteration() > -2) {
        setAlteration(getAlteration() - 1);
    }
}

static int letterToSemitone(char L) {
    static const int s[8] = {0, 0, 2, 4, 5, 7, 9, 11};
    int k = fromLetterToStep(L);
    return s[k];
}

std::string Note::toString(bool omitOctave) const {
    std::ostringstream o;
    o << letter_;
    if (alteration_ == 2) {
        o << 'x';
    }
    if (alteration_ == 1) {
        o << '#';
    }
    if (alteration_ == -1) {
        o << 'b';
    }
    if (alteration_ == -2) {
        o << "bb";
    }
}

```

```

        if (!omitOctave) {
            o << octave_;
        }
        return o.str();
    }

    std::string Note::toStringGerman(bool omitOctave) const {
        std::ostringstream o;
        if (letter_ == 'B' && alteration_ == 0) {
            o << 'h';
        } else if (letter_ == 'B' && alteration_ == 1) {
            o << "his";
        } else if (letter_ == 'B' && alteration_ == -1) {
            o << 'b';
        } else if (letter_ == 'B' && alteration_ == -2) {
            o << "heses";
        } else if (letter_ == 'A' && alteration_ == -1) {
            o << "as";
        } else if (letter_ == 'A' && alteration_ == -2) {
            o << "ases";
        } else if (letter_ == 'E' && alteration_ == -1) {
            o << "es";
        } else if (letter_ == 'E' && alteration_ == -2) {
            o << "eses";
        } else {
            char l =
static_cast<char>(std::tolower(static_cast<unsigned
char>(letter_)));
            o << l;
            if (alteration_ == 2) {
                o << "isis";
            }
            if (alteration_ == 1) {
                o << "is";
            }
            if (alteration_ == -1) {
                o << "es";
            }
            if (alteration_ == -2) {
                o << "eses";
            }
        }
        if (!omitOctave) {
            o << octave_;
        }
        return o.str();
    }

    int Note::getMidi() const {
        int semi = letterToSemitone(letter_) + alteration_;
        return 12 * (octave_ - 4) + 60 + semi;
    }

```

```

bool Note::operator>(const Note &o) const {
    if (octave_ != o.octave_) {
        return octave_ > o.octave_;
    }
    if (letter_ != o.letter_) {
        return fromLetterToStep(letter_) >
fromLetterToStep(o.letter_);
    }
    return alteration_ > o.alteration_;
}

bool Note::operator>=(const Note &o) const { return !(*this < o);
}

bool Note::operator<(const Note &o) const {
    if (octave_ != o.octave_) {
        return octave_ < o.octave_;
    }
    if (letter_ != o.letter_) {
        return fromLetterToStep(letter_) <
fromLetterToStep(o.letter_);
    }
    return alteration_ < o.alteration_;
}

bool Note::operator<=(const Note &o) const { return !(*this > o);
}

bool Note::operator==(const Note &o) const {
    return letter_ == o.letter_ && alteration_ == o.alteration_;
}

bool Note::operator!=(const Note &o) const { return !(*this == o);
}

bool Note::equalsWithOctave(const Note &o) const {
    return letter_ == o.letter_ && alteration_ == o.alteration_
&& octave_ == o.octave_;
}

```

Листинг Б.5 – Interval.h

```

#pragma once

#include "Note.h"
#include <string>

class Interval {
private:
    int index_;
    int type_;
    bool direction_;

public:

```

```

Interval();

void setIndex(int indx);
void setType(int t);
void setDirection(bool d);

int getIndex() const { return index_; }
int getType() const { return type_; }
bool getDirection() const { return direction_; }
int getIntDirection() const { return direction_ ? 1 : -1; }

std::string fromIntervalToString() const;
void formStringToInterval(const std::string &stroke);

void getInterval(Note first, Note second);

bool isItWide() const;

bool operator>(const Interval &o) const;
bool operator>=(const Interval &o) const;
bool operator<(const Interval &o) const;
bool operator<=(const Interval &o) const;
bool operator==(const Interval &o) const;
bool operator!=(const Interval &o) const;
bool equalsWithDirection(const Interval &o) const;
};

```

Листинг Б.6 – Код «Interval.cpp»

```

#include "Interval.h"
#include "Constants.h"
#include "NoteMethods.h"
#include <cstdlib>
#include <sstream>

Interval::Interval() : index_(0), type_(0), direction_(true) {}

void Interval::setIndex(int indx) { index_ = indx; }

void Interval::setType(int t) { type_ = t; }

void Interval::setDirection(bool d) { direction_ = d; }

static const char *typeStr(int type) {
    switch (type) {
        case 2:
            return "aug";
        case 1:
            return "maj";
        case 0:
            return "nat";
        case -1:
            return "min";
        case -2:

```

```

        return "dim";
    default:
        return "nat";
    }
}

static int baseSemitoneDistance(int index) {
    static const int semitones[7] = {0, 2, 4, 5, 7, 9, 11};
    const int simple = ((index % 7) + 7) % 7;
    return (index / 7) * 12 + semitones[simple];
}

static bool isPerfectIntervalIndex(int index) {
    const int simple = ((index % 7) + 7) % 7;
    return simple == 0 || simple == 3 || simple == 4;
}

static int typeFromSemitoneDistance(int index, int
semitoneDistance) {
    const int delta = semitoneDistance -
baseSemitoneDistance(index);
    if (isPerfectIntervalIndex(index)) {
        if (delta > 0) {
            return 2;
        }
        if (delta < 0) {
            return -2;
        }
        return 0;
    }

    if (delta > 0) {
        return 2;
    }
    if (delta == 0) {
        return 1;
    }
    if (delta == -1) {
        return -1;
    }
    return -2;
}

std::string Interval::fromIntervalToString() const {
    std::ostringstream o;
    o << typeStr(type_) << index_;
    if (!(index_ == 0 && type_ == 0)) {
        o << (direction_ ? "U" : "D");
    }
    return o.str();
}

void Interval::formStringToInterval(const std::string &stroke) {

```

```

    if (stroke.size() < 4) {
        return;
    }
    std::string kind = stroke.substr(0, 3);
    size_t p = 3;
    if (p >= stroke.size()) {
        return;
    }
    size_t numStart = p;
    while (p < stroke.size() && stroke[p] >= '0' && stroke[p] <=
'9') {
        p++;
    }
    if (numStart == p) {
        return;
    }
    int idx = std::atoi(stroke.substr(numStart, p -
numStart).c_str());
    setIndex(idx);
    int t = 0;
    if (kind == "aug") {
        t = 2;
    } else if (kind == "maj") {
        t = 1;
    } else if (kind == "nat") {
        t = 0;
    } else if (kind == "min") {
        t = -1;
    } else if (kind == "dim") {
        t = -2;
    }
    type_ = t;
    if (p < stroke.size()) {
        if (stroke[p] == 'U') {
            setDirection(true);
        } else if (stroke[p] == 'D') {
            setDirection(false);
        }
    }
}

void Interval::getInterval(Note first, Note second) {
    int step1 = fromLetterToStep(first.getLetter());
    int step2 = fromLetterToStep(second.getLetter());
    int diat = (second.getOctave() - first.getOctave()) * 7 +
(step2 - step1);
    bool up = diat >= 0;
    if (diat == 0 && first.getMidi() > second.getMidi()) {
        up = false;
    }
    int absD = std::abs(diat);
    setIndex(absD);
    setDirection(up);
}

```

```

        type_ = typeFromSemitoneDistance(absD,
std::abs(second.getMidi() - first.getMidi()));
    }

bool Interval::isItWide() const { return index_ > 4; }

bool Interval::operator>(const Interval &o) const {
    if (index_ != o.index_) {
        return index_ > o.index_;
    }
    return type_ > o.type_;
}

bool Interval::operator>=(const Interval &o) const {
    if (index_ != o.index_) {
        return index_ >= o.index_;
    }
    return type_ >= o.type_;
}

bool Interval::operator<(const Interval &o) const {
    if (index_ != o.index_) {
        return index_ < o.index_;
    }
    return type_ < o.type_;
}

bool Interval::operator<=(const Interval &o) const {
    if (index_ != o.index_) {
        return index_ <= o.index_;
    }
    return type_ <= o.type_;
}

bool Interval::operator==(const Interval &o) const {
    return index_ == o.index_ && type_ == o.type_;
}

bool Interval::operator!=(const Interval &o) const { return
!(*this == o); }

bool Interval::equalsWithDirection(const Interval &o) const {
    return index_ == o.index_ && type_ == o.type_ && direction_
== o.direction_;
}

```

Листинг Б.7 – Код «Constants.h»

```

#pragma once

#include "Note.h"
#include <map>
#include <set>

```



```

        "проходившего обучение по направлению
подготовки 09.03.02 «Информационные системы и "
        "технологии» (с 2021 года);\n    2) отделения
«Теория музыки» Пензенского музыкального"
        " училища (ныне колледжа) имени Александра
Андреевича Архангельского (с 2022 года).\n\n"
        "\t\t\t\t\t2026 год";

```

Листинг Б.9 – Код «Globals.h»

```

#pragma once

#include "Interval.h"
#include "Note.h"
#include <string>

constexpr int SERIA_COUNT = 12;
constexpr int INTERVAL_COUNT = 11;
constexpr int MATRIX_CELLS = 144;

extern Note thisSeria[SERIA_COUNT];
extern Interval thisIntervals[INTERVAL_COUNT];
extern Note thisMatrix[MATRIX_CELLS];

extern int alalysResult;
extern bool isSeriaInitialized;
extern int analysedSeriaSwitch;
extern std::string analysResultMessage;

void resetSeriaState();

```

Листинг Б.10 – Код «Globals.cpp»

```

#include "Globals.h"

Note thisSeria[SERIA_COUNT];
Interval thisIntervals[INTERVAL_COUNT];
Note thisMatrix[MATRIX_CELLS];

int alalysResult = 0;
bool isSeriaInitialized = false;
int analysedSeriaSwitch = 0;
std::string analysResultMessage;

void resetSeriaState() {
    for (int i = 0; i < SERIA_COUNT; ++i) {
        thisSeria[i] = Note();
    }
    for (int i = 0; i < INTERVAL_COUNT; ++i) {
        thisIntervals[i] = Interval();
    }
    for (int i = 0; i < MATRIX_CELLS; ++i) {
        thisMatrix[i] = Note();
    }
}

```

```

        alalysResult = 0;
        isSeriaInitialized = false;
        analysResultMessage.clear();
    }

```

Листинг Б.11 – Код «NoteMethods.h»

```

#pragma once

#include "Interval.h"
#include "Note.h"
#include <string>

char fromStepToLetter(int step);
int fromLetterToStep(char L);
std::string getNaturalInterval(char base, char soprano);
Note getNewNoteFromInterval(Note first, Interval intrvl);

```

Листинг Б.12 – Код «NoteMethods.cpp»

```

#include "NoteMethods.h"
#include "Constants.h"

char fromStepToLetter(int step) {
    int n = ((step - 1) % 7 + 7) % 7 + 1;
    const char *map = " CDEFGAB";
    return map[n];
}

int fromLetterToStep(char L) {
    switch (L) {
        case 'C':
            return 1;
        case 'D':
            return 2;
        case 'E':
            return 3;
        case 'F':
            return 4;
        case 'G':
            return 5;
        case 'A':
            return 6;
        case 'B':
            return 7;
        default:
            return 1;
    }
}

std::string getNaturalInterval(char base, char soprano) {
    if ((base == 'E' && soprano == 'F') || (base == 'F' &&
soprano == 'E')) {
        return "min1";
    }
}

```

```

    }
    if ((base == 'B' && soprano == 'C') || (base == 'C' &&
soprano == 'B')) {
        return "min1";
    }
    if ((base == 'C' && soprano == 'D') || (base == 'D' &&
soprano == 'C')) {
        return "maj1";
    }
    if ((base == 'D' && soprano == 'E') || (base == 'E' &&
soprano == 'D')) {
        return "maj1";
    }
    if ((base == 'F' && soprano == 'G') || (base == 'G' &&
soprano == 'F')) {
        return "maj1";
    }
    if ((base == 'G' && soprano == 'A') || (base == 'A' &&
soprano == 'G')) {
        return "maj1";
    }
    if ((base == 'A' && soprano == 'B') || (base == 'B' &&
soprano == 'A')) {
        return "maj1";
    }
    if ((base == 'D' && soprano == 'F') || (base == 'F' &&
soprano == 'D')) {
        return "min2";
    }
    if ((base == 'E' && soprano == 'G') || (base == 'G' &&
soprano == 'E')) {
        return "min2";
    }
    if ((base == 'A' && soprano == 'C') || (base == 'C' &&
soprano == 'A')) {
        return "min2";
    }
    if ((base == 'B' && soprano == 'D') || (base == 'D' &&
soprano == 'B')) {
        return "min2";
    }
    if ((base == 'C' && soprano == 'E') || (base == 'E' &&
soprano == 'C')) {
        return "maj2";
    }
    if ((base == 'F' && soprano == 'A') || (base == 'A' &&
soprano == 'F')) {
        return "maj2";
    }
    if ((base == 'G' && soprano == 'B') || (base == 'B' &&
soprano == 'G')) {
        return "maj2";
    }
}

```

```

        if ((base == 'C' && soprano == 'F') || (base == 'F' &&
soprano == 'C')) {
            return "nat3";
        }
        if (base == 'C' && soprano == 'G') {
            return "nat4";
        }
        if (base == 'G' && soprano == 'C') {
            return "nat3";
        }
        if ((base == 'D' && soprano == 'G') || (base == 'G' &&
soprano == 'D')) {
            return "nat3";
        }
        if ((base == 'D' && soprano == 'A') || (base == 'A' &&
soprano == 'D')) {
            return "nat4";
        }
        if ((base == 'E' && soprano == 'A') || (base == 'A' &&
soprano == 'E')) {
            return "nat3";
        }
        if ((base == 'E' && soprano == 'B') || (base == 'B' &&
soprano == 'E')) {
            return "nat4";
        }
        if ((base == 'F' && soprano == 'C') || (base == 'C' &&
soprano == 'F')) {
            return "nat4";
        }
        if ((base == 'G' && soprano == 'D') || (base == 'D' &&
soprano == 'G')) {
            return "nat4";
        }
        if ((base == 'A' && soprano == 'D') || (base == 'D' &&
soprano == 'A')) {
            return "nat3";
        }
        if ((base == 'A' && soprano == 'E') || (base == 'E' &&
soprano == 'A')) {
            return "nat4";
        }
        if ((base == 'B' && soprano == 'E') || (base == 'E' &&
soprano == 'B')) {
            return "nat3";
        }
        if ((base == 'E' && soprano == 'C') || (base == 'C' &&
soprano == 'E')) {
            return "min5";
        }
        if ((base == 'A' && soprano == 'F') || (base == 'F' &&
soprano == 'A')) {
            return "min5";
        }

```

```

    }
    if ((base == 'B' && soprano == 'G') || (base == 'G' &&
soprano == 'B')) {
        return "min5";
    }
    if ((base == 'C' && soprano == 'A') || (base == 'A' &&
soprano == 'C')) {
        return "maj5";
    }
    if ((base == 'D' && soprano == 'B') || (base == 'B' &&
soprano == 'D')) {
        return "maj5";
    }
    if ((base == 'F' && soprano == 'D') || (base == 'D' &&
soprano == 'F')) {
        return "maj5";
    }
    if ((base == 'G' && soprano == 'E') || (base == 'E' &&
soprano == 'G')) {
        return "maj5";
    }
    if ((base == 'D' && soprano == 'B') || (base == 'B' &&
soprano == 'D')) {
        return "min6";
    }
    if ((base == 'E' && soprano == 'D') || (base == 'D' &&
soprano == 'E')) {
        return "min6";
    }
    if ((base == 'G' && soprano == 'F') || (base == 'F' &&
soprano == 'G')) {
        return "min6";
    }
    if ((base == 'A' && soprano == 'G') || (base == 'G' &&
soprano == 'A')) {
        return "min6";
    }
    if ((base == 'B' && soprano == 'A') || (base == 'A' &&
soprano == 'B')) {
        return "min6";
    }
    if ((base == 'C' && soprano == 'B') || (base == 'B' &&
soprano == 'C')) {
        return "maj6";
    }
    if ((base == 'F' && soprano == 'E') || (base == 'E' &&
soprano == 'F')) {
        return "maj6";
    }
    if ((base == 'B' && soprano == 'F') || (base == 'F' &&
soprano == 'B')) {
        return "dim4";
    }
}

```

```

        if ((base == 'F' && soprano == 'B') || (base == 'B' &&
soprano == 'F')) {
            return "aug3";
        }
        return "nat3";
    }

static bool isPerfectIntervalIndex(int index) {
    const int simple = ((index % 7) + 7) % 7;
    return simple == 0 || simple == 3 || simple == 4;
}

static int baseSemitoneDistance(int index) {
    static const int semitones[7] = {0, 2, 4, 5, 7, 9, 11};
    const int simple = ((index % 7) + 7) % 7;
    return (index / 7) * 12 + semitones[simple];
}

static int intervalSemitoneDistance(const Interval &intrvl) {
    int distance = baseSemitoneDistance(intrvl.getIndex());
    const int type = intrvl.getType();
    if (isPerfectIntervalIndex(intrvl.getIndex())) {
        if (type == 2) {
            ++distance;
        } else if (type == -2) {
            --distance;
        }
    } else {
        if (type == -1) {
            --distance;
        } else if (type == 2) {
            ++distance;
        } else if (type == -2) {
            distance -= 2;
        }
    }
    return distance < 0 ? 0 : distance;
}

Note getNewNoteFromInterval(Note first, Interval intrvl) {
    Note second;
    int dir = intrvl.getIntDirection();
    int idx = intrvl.getIndex();
    int step1 = fromLetterToStep(first.getLetter());
    int step2 = step1 + dir * idx;
    int oct = first.getOctave();
    while (step2 > 7) {
        step2 -= 7;
        oct++;
    }
    while (step2 < 1) {
        step2 += 7;
        oct--;
    }
}

```

```

    }
    second.setLetter(fromStepToLetter(step2));
    second.setOctave(oct);

    const int targetMidi = first.getMidi() + dir *
intervalSemitoneDistance(intrvl);
    const int naturalMidi = Note(second.getLetter(), 0,
second.getOctave()).getMidi();
    const int secondAlt = targetMidi - naturalMidi;
    if (secondAlt >= -2 && secondAlt <= 2) {
        second.setAlteration(secondAlt);
    } else {
        second.setAlteration(secondAlt > 0 ? 2 : -2);
    }
    return second;
}

```

Листинг Б.13 – Код «SeriaMethods.h»

```

#pragma once

#include "Interval.h"
#include "Note.h"
#include <string>

struct SeriaMethods {
    static Interval localIntervalEnharmonicChange(const Interval
&in);

    static void collectIntervals(const Note *seria, Interval
*outIntervals);

    static void primusSeria(const Note &key, const Interval
*primusIntervals, Note *outSeria);

    static void retroversusSeria(const Note *seria, Note
*outSeria);

    static void inversusSeria(const Note *seria, const Interval
*seriaIntervals, Note *outSeria);

    static void retroinversusSeria(const Note *seria, const
Interval *seriaIntervals, Note *outSeria);

    static void buildMatrix(const Note *primus, Note
*matrixRowMajor144);

    static bool rule1(const Interval *iv, const Note *s, int n);
    static bool rule2(const Note *s);
    static bool rule4(const Note *s);
    static bool rule5(const Interval *iv, int n);
    static bool rule6(const Interval *iv, int n);
    static bool rule7(const Interval *iv, int n);
    static bool rule8(const Interval *iv, int n);

```

```

        static void runAnalysis(std::string &msg, int &code);

        static std::string matrixCellToString(const Note &n);
};

```

Листинг Б.14 – Код «SeriaMethods.cpp»

```

#include "SeriaMethods.h"
#include "Constants.h"
#include "Globals.h"
#include "NoteMethods.h"
#include <algorithm>
#include <cstdlib>
#include <cstring>

Interval SeriaMethods::localIntervalEnharmonicChange(const
Interval &in) {
    Interval o = in;
    const int ix = in.getIndex();
    const int ty = in.getType();
    if (ix == 3 && ty == -2) {
        o.setIndex(2);
        o.setType(1);
        return o;
    }
    if (ix == 4 && ty == 2) {
        o.setIndex(5);
        o.setType(1);
        return o;
    }
    if (ix == 2 && ty == -2) {
        o.setIndex(1);
        o.setType(1);
        return o;
    }
    if (ix == 2 && ty == 2) {
        o.setIndex(3);
        o.setType(0);
        return o;
    }
    if (ix == 5 && ty == 2) {
        o.setIndex(6);
        o.setType(1);
        return o;
    }
    if (ix == 5 && ty == -2) {
        o.setIndex(4);
        o.setType(0);
        return o;
    }
    return o;
}

```

```

void SeriaMethods::collectIntervals(const Note *seria, Interval
*outIntervals) {
    for (int i = 0; i < INTERVAL_COUNT; ++i) {
        outIntervals[i].getInterval(seria[i], seria[i + 1]);
    }
}

static void clampIntoSystemRange(Note &n) {
    int guard = 0;
    while (n > NOTE_HIGHEST_TONE && n.getOctave() > 0 && guard <
24) {
        n.setOctave(n.getOctave() - 1);
        ++guard;
    }
    guard = 0;
    while (n < NOTE_LOWEST_TONE && n.getOctave() < 7 && guard <
24) {
        n.setOctave(n.getOctave() + 1);
        ++guard;
    }
}

void SeriaMethods::primusSeria(const Note &key, const Interval
*primusIntervals, Note *outSeria) {
    outSeria[0] = key;
    clampIntoSystemRange(outSeria[0]);
    for (int i = 0; i < INTERVAL_COUNT; ++i) {
        outSeria[i + 1] = getNewNoteFromInterval(outSeria[i],
primusIntervals[i]);
        clampIntoSystemRange(outSeria[i + 1]);
    }
}

void SeriaMethods::retroversusSeria(const Note *seria, Note
*outSeria) {
    for (int i = 0; i < SERIA_COUNT; ++i) {
        outSeria[i] = seria[SERIA_COUNT - 1 - i];
    }
}

static void makeInversusIntervals(const Interval *seriaIntervals,
Interval *outIntervals) {
    for (int i = 0; i < INTERVAL_COUNT; ++i) {
        outIntervals[i] = seriaIntervals[i];
        outIntervals[i].setDirection(!outIntervals[i].getDirectio
n());
    }
}

void SeriaMethods::inversusSeria(const Note *seria, const Interval
*seriaIntervals, Note *outSeria) {
    Interval invIntervals[INTERVAL_COUNT];
    makeInversusIntervals(seriaIntervals, invIntervals);
}

```

```

        primusSeria(seria[0], invIntervals, outSeria);
    }

void SerialMethods::retroinversusSeria(const Note *seria, const
Interval *seriaIntervals, Note *outSeria) {
    Note tmp[SERIA_COUNT];
    inversusSeria(seria, seriaIntervals, tmp);
    retroversusSeria(tmp, outSeria);
}

void SerialMethods::buildMatrix(const Note *primus, Note
*matrixRowMajor144) {
    Interval invIntervals[INTERVAL_COUNT];
    makeInversusIntervals(thisIntervals, invIntervals);

    for (int c = 0; c < SERIA_COUNT; ++c) {
        Note column[SERIA_COUNT];
        primusSeria(primus[c], invIntervals, column);
        for (int r = 0; r < SERIA_COUNT; ++r) {
            matrixRowMajor144[r * SERIA_COUNT + c] = column[r];
        }
    }

    for (int c = 0; c < SERIA_COUNT; ++c) {
        matrixRowMajor144[c] = primus[c];
    }
}

static bool allSemitoneSteps(const Note *s, int n) {
    for (int i = 0; i < n; ++i) {
        if (s[i + 1].getMidi() - s[i].getMidi() != 1) {
            return false;
        }
    }
    return true;
}

bool SerialMethods::rule1(const Interval *iv, const Note *s, int n)
{
    if (s && allSemitoneSteps(s, n)) {
        return false;
    }
    bool allMinSeconds = true;
    for (int i = 0; i < n; ++i) {
        if (!(iv[i].getIndex() == 1 && iv[i].getType() == -1)) {
            allMinSeconds = false;
            break;
        }
    }
    if (allMinSeconds) {
        return false;
    }
    bool foundImperfect = false;

```

```

    for (int i = 0; i < n; ++i) {
        if (!PERFECT_CONSONANCES.count(iv[i].getIndex())) {
            foundImperfect = true;
            break;
        }
    }
    if (!foundImperfect) {
        return false;
    }
    return true;
}

bool SerialMethods::rule2(const Note *s) {
    for (int i = 0; i < SERIA_COUNT; ++i) {
        for (int j = i + 1; j < SERIA_COUNT; ++j) {
            if (s[i] == s[j]) {
                return false;
            }
        }
    }
    return true;
}

bool SerialMethods::rule4(const Note *s) {
    Note sorted[SERIA_COUNT];
    for (int i = 0; i < SERIA_COUNT; ++i) {
        sorted[i] = s[i];
    }

    for (int i = 0; i < SERIA_COUNT - 1; ++i) {
        for (int j = 0; j < SERIA_COUNT - i - 1; ++j) {
            if (sorted[j + 1] < sorted[j]) {
                const Note tmp = sorted[j];
                sorted[j] = sorted[j + 1];
                sorted[j + 1] = tmp;
            }
        }
    }

    Interval span;
    span.getInterval(sorted[0], sorted[SERIA_COUNT - 1]);
    if (span.getIndex() >= 9 && span.getType() > 0) {
        return false;
    }
    return true;
}

bool SerialMethods::rule5(const Interval *iv, int n) {
    for (int i = 0; i < n; ++i) {
        if (iv[i].getIndex() > 7) {
            return false;
        }
    }
}

```

```

        return true;
    }

    bool SeriaMethods::rule6(const Interval *iv, int n) {
        if (n < 3) {
            return true;
        }
        for (int i = 0; i <= n - 3; ++i) {
            if (iv[i] == iv[i + 1] && iv[i + 1] == iv[i + 2]) {
                return false;
            }
        }
        return true;
    }

    bool SeriaMethods::rule7(const Interval *iv, int n) {
        if (n < 3) {
            return true;
        }
        for (int i = 0; i <= n - 3; ++i) {
            const Interval &a = iv[i];
            const Interval &b = iv[i + 1];
            const Interval &c = iv[i + 2];
            const bool aWide = a.isItWide();
            const bool cWide = c.isItWide();
            const bool aTight = a.getIndex() < 4;
            const bool cTight = c.getIndex() < 4;
            if (aWide && cTight) {
                return false;
            }
            if (aTight && cWide) {
                return false;
            }
            const bool sameKind = (aWide && cWide) || (aTight &&
cTight);
            if (sameKind && b == a) {
                return false;
            }
        }
        return true;
    }

    bool SeriaMethods::rule8(const Interval *, int) { return true; }

    void SeriaMethods::runAnalysis(std::string &msg, int &code) {
        msg.clear();
        const bool r1 = rule1(thisIntervals, thisSeria,
INTERVAL_COUNT);
        if (!r1) {
            msg += u8"\na) серия не должна быть тождественна
хроматической гамме, квартовому или квинтовому кругу";
        }
        const bool r2 = rule2(thisSeria);

```

```

    if (!r2) {
        msg += u8"\nб) каждый тон в этом схематическом
образовании появляется только однажды";
    }
    const bool r4 = rule4(thisSeria);
    if (!r4) {
        msg += u8"\nг) серия не должна слишком выходить за рамки
диапазона октавы или ноты (ваша вышла за "
        u8"большую квартдециму)";
    }
    const bool r5 = rule5(thisIntervals, INTERVAL_COUNT);
    if (!r5) {
        msg += u8"\нд) скачки больше чем на октаву, учитывая
возможные октавные транспозиции тонов, не имеют "
        u8"смысла";
    }
    const bool r6 = rule6(thisIntervals, INTERVAL_COUNT);
    if (!r6) {
        msg += u8"\не) не больше двух одинаковых интервалов могут
следовать подряд друг за другом";
    }
    const bool r7 = rule7(thisIntervals, INTERVAL_COUNT);
    if (!r7) {
        msg += u8"\нж) малые и большие (широкие) интервалы лучше
чередовать";
    }
    const bool r8 = rule8(thisIntervals, INTERVAL_COUNT);
    if (!r8) {
        msg += u8"\нз) не допускаются мелодические ходы,
создающие арпеджио диатонических аккордов (особенно "
        u8"трезвучий)";
    }

    if (!r1 || !r2 || !r5) {
        code = -1;
    } else if (!r4) {
        code = 1;
    } else if (!r6 || !r7 || !r8) {
        code = 2;
    } else {
        code = 3;
    }

    if (code <= 0) {
        msg += u8"\n\nРезультат - провальный. Построить матрицу
невозможно.";
    } else if (code == 1) {
        msg +=
            u8"\n\nРезультат - успешный, однако вывести формы
будет невозможно (слишком широкий диапазон серии).";
    } else if (code == 2) {
        msg += u8"\n\nРезультат - успешный.";
    } else {

```

```

        msg += u8"\n\nРезультат - успешный! Серия
ортодоксальная.";
    }
}

std::string SeriaMethods::matrixCellToString(const Note &n) {
    Note d = n;
    if (d.getAlteration() == 2 || d.getAlteration() == -2) {
        d.enharmonicChange();
    }
    return d.toStringGerman(true);
}

```

Листинг Б.15 – Код «StaffRenderer.h»

```

#pragma once

#include "Globals.h"
#include <string>

namespace StaffRender {
    constexpr int STAFF_ROWS = 21;
    constexpr int NOTE_SLOT_WIDTH = 8;
    constexpr int STAFF_COLS = SERIA_COUNT * NOTE_SLOT_WIDTH;

    bool isStaffLineRow(int row1Based);
    int staffRowFromNote(const Note &note);
    Note noteFromStaffRow(int row1Based);
    void stepPitchUp(Note &note);
    void stepPitchDown(Note &note);
    std::string renderStaffUtf8(const Note notes[SERIA_COUNT],
int selectedIndex = -1);
}

```

Листинг Б.16 – Код «StaffRenderer.cpp»

```

#include "StaffRender.h"

#include "Constants.h"

#include "NoteMethods.h"

#include <cstring>

namespace StaffRender {
    namespace {
        constexpr int kMainStaffLines[] = {8, 10, 12, 14, 16};
        constexpr int kLedgerAbove[] = {2, 4, 6};
        constexpr int kLedgerBelow[] = {18, 20};
        constexpr int kMainTop = 8;
        constexpr int kMainBottom = 16;
        constexpr int kMidiRef = 60;

        struct PitchSpelling {

```

```

        char letter;
        int alteration;
    };

    constexpr PitchSpelling kUpSpelling[] = {
        {'C', 0}, {'C', 1}, {'D', 0}, {'D', 1}, {'E', 0},
{'F', 0},
        {'F', 1}, {'G', 0}, {'G', 1}, {'A', 0}, {'A', 1},
{'B', 0},
    };

    constexpr PitchSpelling kDownSpelling[] = {
        {'C', 0}, {'D', -1}, {'D', 0}, {'E', -1}, {'E', 0},
{'F', 0},
        {'G', -1}, {'G', 0}, {'A', -1}, {'A', 0}, {'B', -1},
{'B', 0},
    };

    int staffMidiMin() { return NOTE_LOWEST_TONE.getMidi(); }
    int staffMidiMax() { return NOTE_HIGHEST_TONE.getMidi(); }
}

    int letterToSemitone(char letter) {
        static const int semitones[8] = {0, 0, 2, 4, 5, 7,
9, 11};
        return semitones[fromLetterToStep(letter)];
    }

    int octaveFromMidi(int midi, char letter, int alteration)
    {
        const int semitone = letterToSemitone(letter) +
alteration;
        int octave = 4 + (midi - kMidiRef - semitone) / 12;
        if (octave < 0) {
            octave = 0;
        }
        if (octave > 7) {
            octave = 7;
        }
        return octave;
    }

    Note noteFromMidiGerman(int midi, bool ascendingSpelling)
    {
        const int pitchClass = ((midi % 12) + 12) % 12;
        const PitchSpelling &spelling = ascendingSpelling ?
kUpSpelling[pitchClass] : kDownSpelling[pitchClass];
        return Note(spelling.letter, spelling.alteration,
                    octaveFromMidi(midi, spelling.letter,
spelling.alteration));
    }

```

```

int diatonicStepsFromG3(const Note &note) {
    const int gStep = fromLetterToStep('G');
    int steps = (note.getOctave() - 3) * 7 +
(fromLetterToStep(note.getLetter()) - gStep);
    if (steps < 0) {
        return 0;
    }
    if (steps > 20) {
        return 20;
    }
    return steps;
}

Note noteFromDiatonicIndex(int index) {
    Note note('G', 0, 3);
    for (int i = 0; i < index; ++i) {
        int step = fromLetterToStep(note.getLetter()) +
1;

        int octave = note.getOctave();
        if (step > 7) {
            step = 1;
            ++octave;
        }
        note.setLetter(fromStepToLetter(step));
        note.setOctave(octave);
    }
    return note;
}

int displayRowFromNote(const Note &note) { return
STAFF_ROWS - diatonicStepsFromG3(note); }

bool isMainStaffLineRow(int row1Based) {
    for (int lineRow : kMainStaffLines) {
        if (lineRow == row1Based) {
            return true;
        }
    }
    return false;
}

bool isLedgerLineRow(int row1Based) {
    for (int lineRow : kLedgerAbove) {
        if (lineRow == row1Based) {
            return true;
        }
    }
    for (int lineRow : kLedgerBelow) {
        if (lineRow == row1Based) {
            return true;
        }
    }
    return false;
}

```

```

    }

    std::string alterationField(const Note &note) {
        switch (note.getAlteration()) {
            case 2:
                return "##";
            case 1:
                return "# ";
            case -1:
                return "b ";
            case -2:
                return "bb";
            default:
                return " ";
        }
    }

    void drawLedgeSegment(std::string &line, int colStart) {
        for (int offset = 0; offset < NOTE_SLOT_WIDTH;
++offset) {
            const size_t idx = static_cast<size_t>(colStart
+ offset);
            if (idx < line.size()) {
                line[idx] = '-';
            }
        }
    }

    void drawLedgeLinesForNote(std::string rows[STAFF_ROWS],
int noteIndex, int noteRow) {
        const int col = noteIndex * NOTE_SLOT_WIDTH;

        if (noteRow < kMainTop) {
            for (int ledgerRow : kLedgerAbove) {
                if (ledgerRow > noteRow && ledgerRow <
kMainTop) {
                    drawLedgeSegment(rows[ledgerRow -
1], col);
                }
            }
        } else if (noteRow > kMainBottom) {
            for (int ledgerRow : kLedgerBelow) {
                if (ledgerRow > kMainBottom && ledgerRow <
noteRow) {
                    drawLedgeSegment(rows[ledgerRow -
1], col);
                }
            }
        }

        if (noteRow < kMainTop || noteRow > kMainBottom) {
            if (isLedgerLineRow(noteRow)) {
                drawLedgeSegment(rows[noteRow - 1], col);
            }
        }
    }

```

```

    }
}

void fillMainStaffLines(std::string rows[STAFF_ROWS]) {
    for (int row = 1; row <= STAFF_ROWS; ++row) {
        rows[row - 1].assign(STAFF_COLS, ' ');
    }
    for (int lineRow : kMainStaffLines) {
        std::string &line = rows[lineRow - 1];
        for (int col = 0; col < STAFF_COLS; ++col) {
            line[static_cast<size_t>(col)] = '-';
        }
    }
}

void placeNote(std::string rows[STAFF_ROWS], int
noteIndex, const Note &note, int selectedIndex) {
    const int row = displayRowFromNote(note);
    const int col = noteIndex * NOTE_SLOT_WIDTH;
    if (row < 1 || row > STAFF_ROWS || col +
NOTE_SLOT_WIDTH > STAFF_COLS) {
        return;
    }

    drawLedgerLinesForNote(rows, noteIndex, row);

    const std::string alt = alterationField(note);
    const char head = (noteIndex == selectedIndex) ? '@'
: 'o';

    std::string &line = rows[row - 1];
    line[static_cast<size_t>(col)] = alt[0];
    line[static_cast<size_t>(col + 1)] = alt[1];
    line[static_cast<size_t>(col + 2)] = head;
    line[static_cast<size_t>(col + 3)] = ' ';
}

} // namespace

bool isStaffLineRow(int row1Based) { return
isMainStaffLineRow(row1Based); }

int staffRowFromNote(const Note &note) { return
diatonicStepsFromG3(note) + 1; }

Note noteFromStaffRow(int row1Based) {
    int row = row1Based;
    if (row < 1) {
        row = 1;
    }
    if (row > STAFF_ROWS) {
        row = STAFF_ROWS;
    }
}

```

```

        return noteFromDiatonicIndex(row - 1);
    }

    void stepPitchUp(Note &note) {
        if (note.getMidi() >= staffMidiMax()) {
            return;
        }
        note = noteFromMidiGerman(note.getMidi() + 1, true);
    }

    void stepPitchDown(Note &note) {
        if (note.getMidi() <= staffMidiMin()) {
            return;
        }
        note = noteFromMidiGerman(note.getMidi() - 1, false);
    }

    std::string renderStaffUtf8(const Note notes[SERIA_COUNT],
int selectedIndex) {
        std::string rows[STAFF_ROWS];
        fillMainStaffLines(rows);

        for (int i = 0; i < SERIA_COUNT; ++i) {
            placeNote(rows, i, notes[i], selectedIndex);
        }

        std::string out;
        for (int row = 0; row < STAFF_ROWS; ++row) {
            out += rows[row];
            if (row + 1 < STAFF_ROWS) {
                out += '\n';
            }
        }

        out += "\n";
        out += "g3";
        {
            const int pad = STAFF_COLS - 2 -
static_cast<int>(std::strlen("fis6"));
            if (pad > 0) {
                out.append(static_cast<size_t>(pad), ' ');
            } else {
                out += ' ';
            }
        }
        out += "fis6";
        out += "\n\n";
        for (int i = 0; i < SERIA_COUNT; ++i) {
            const std::string label =
notes[i].toStringGerman(false);
            out += label;
            const int pad = NOTE_SLOT_WIDTH -
static_cast<int>(label.size());

```

```

        if (pad > 0) {
            out.append(static_cast<size_t>(pad), ' ');
        } else if (pad < 0) {
            out.append(1, ' ');
        }
    }

    return out;
}
}

```

Листинг Б.17 – Код «FileMethods.h»

```

#pragma once

#include "Note.h"
#include <string>

bool saveSeriaToFile(const std::string &dirPath, const std::string
&fileName, const Note *seria, int count);

bool saveTextToFile(const std::string &dirPath, const std::string
&fileName, const std::string &utf8Content);

bool loadSeriaFromFile(const std::string &dirPath, const
std::string &fileName, Note *seria, int count);

```

Листинг Б.18 – Код «FileMethods.cpp»

```

#include "FileMethods.h"
#include "Constants.h"
#include "Encoding.h"
#include <cctype>
#include <filesystem>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>

namespace {
    std::string trim(const std::string &s) {
        size_t a = 0;
        while (a < s.size() && std::isspace(static_cast<unsigned
char>(s[a]))) {
            ++a;
        }
        size_t b = s.size();
        while (b > a && std::isspace(static_cast<unsigned
char>(s[b - 1]))) {
            --b;
        }
        return s.substr(a, b - a);
    }
}

```

```

    std::string toLowerAscii(std::string s) {
        for (char &ch : s) {
            ch =
static_cast<char>(std::tolower(static_cast<unsigned char>(ch)));
        }
        return s;
    }

    struct NoteNamePattern {
        const char *name;
        char letter;
        int alteration;
    };

    bool parseGermanNoteName(const std::string &raw, Note &n) {
        std::string token = toLowerAscii(trim(raw));
        if (token.empty()) {
            return false;
        }

        int octave = 4;
        size_t nameEnd = token.size();
        while (nameEnd > 0 && std::isdigit(static_cast<unsigned
char>(token[nameEnd - 1]))) {
            --nameEnd;
        }
        if (nameEnd < token.size()) {
            if (nameEnd + 1 != token.size()) {
                return false;
            }
            octave = token[nameEnd] - '0';
            if (octave < 0 || octave > 7) {
                return false;
            }
            token = token.substr(0, nameEnd);
        }
        if (token.empty()) {
            return false;
        }

        static const NoteNamePattern kPatterns[] = {
            {"heses", 'B', -2}, {"ases", 'A', -2}, {"eses", 'E',
-2}, {"deses", 'D', -2},
            {"cisis", 'C', 2}, {"disis", 'D', 2}, {"fisis",
'F', 2}, {"gisis", 'G', 2},
            {"aisis", 'A', 2},
            {"cis", 'C', 1}, {"dis", 'D', 1}, {"fis", 'F',
1}, {"gis", 'G', 1},
            {"ais", 'A', 1},
            {"ces", 'C', -1}, {"des", 'D', -1}, {"ges", 'G',
-1}, {"as", 'A', -1},
            {"es", 'E', -1},
            {"h", 'B', 0}, {"b", 'B', -1},

```

```

        {"c", 'C', 0},        {"d", 'D', 0},        {"e", 'E',
0},
        {"f", 'F', 0},        {"g", 'G', 0},        {"a", 'A',
0},
    };

    for (const NoteNamePattern &pattern : kPatterns) {
        if (token == pattern.name) {
            n = Note(pattern.letter, pattern.alteration,
octave);
            return true;
        }
    }
    return false;
}

bool parseAmericanNoteName(const std::string &raw, Note &n) {
    std::string token = trim(raw);
    if (token.empty()) {
        return false;
    }

    size_t i = 0;
    char letter =
static_cast<char>(std::toupper(static_cast<unsigned
char>(token[i++])));
    if (!NOTE_LETTERS.count(letter)) {
        return false;
    }

    int alteration = 0;
    if (i < token.size() && token[i] == 'x') {
        alteration = 2;
        ++i;
    } else if (i < token.size() && token[i] == '#') {
        alteration = 1;
        ++i;
    } else if (i + 1 < token.size() && token[i] == 'b' &&
token[i + 1] == 'b') {
        alteration = -2;
        i += 2;
    } else if (i < token.size() && token[i] == 'b') {
        alteration = -1;
        ++i;
    }

    int octave = 4;
    if (i < token.size() && std::isdigit(static_cast<unsigned
char>(token[i]))) {
        octave = token[i] - '0';
        if (octave < 0 || octave > 7) {
            return false;
        }
    }

```

```

    }

    n = Note(letter, alteration, octave);
    return true;
}

bool parseNoteToken(const std::string &raw, Note &n) {
    if (parseGermanNoteName(raw, n)) {
        return true;
    }
    return parseAmericanNoteName(raw, n);
}

std::vector<std::string> splitNoteTokens(const std::string
&content) {
    std::vector<std::string> tokens;
    std::string current;

    for (size_t i = 0; i < content.size(); ++i) {
        const char ch = content[i];
        if (ch == '\n' || ch == '\r' || ch == '\t' || ch ==
' ') {
            const std::string piece = trim(current);
            if (!piece.empty()) {
                tokens.push_back(piece);
            }
            current.clear();
            continue;
        }
        current += ch;
    }

    const std::string piece = trim(current);
    if (!piece.empty()) {
        tokens.push_back(piece);
    }
    return tokens;
}

std::string joinPath(const std::string &dirPath, const
std::string &fileName) {
    if (dirPath.empty()) {
        return fileName;
    }
    if (dirPath.back() == '\\\' || dirPath.back() == '/') {
        return dirPath + fileName;
    }
    return dirPath + '\\\' + fileName;
}

std::filesystem::path pathFromUtf8(const std::string &utf8) {
#ifdef _WIN32
    return std::filesystem::path(utf8ToWide(utf8));

```

```

        #else
            return std::filesystem::path(utf8);
        #endif
    }

    std::string readFileUtf8(const std::string &path) {
        std::ifstream in(pathFromUtf8(path), std::ios::in |
std::ios::binary);
        if (!in) {
            return {};
        }
        std::string content((std::istreambuf_iterator<char>(in)),
std::istreambuf_iterator<char>());
        if (content.size() >= 3 &&
            static_cast<unsigned char>(content[0]) == 0xEF &&
            static_cast<unsigned char>(content[1]) == 0xBB &&
            static_cast<unsigned char>(content[2]) == 0xBF) {
            content.erase(0, 3);
        }
        return content;
    }
}

bool saveSeriaToFile(const std::string &dirPath, const std::string
&fileName, const Note *seria, int count) {
    const std::string path = joinPath(dirPath, fileName);
    std::ofstream out(pathFromUtf8(path), std::ios::out |
std::ios::trunc);
    if (!out) {
        return false;
    }
    for (int k = 0; k < count; ++k) {
        if (k > 0) {
            out << ' ';
        }
        out << seria[k].toStringGerman(false);
    }
    out << '\n';
    return static_cast<bool>(out);
}

bool saveTextToFile(const std::string &dirPath, const std::string
&fileName, const std::string &utf8Content) {
    const std::string path = joinPath(dirPath, fileName);
    std::ofstream out(pathFromUtf8(path), std::ios::out |
std::ios::trunc);
    if (!out) {
        return false;
    }
    out << utf8Content;
    if (!utf8Content.empty() && utf8Content.back() != '\n') {
        out << '\n';
    }
}

```

```

        return static_cast<bool>(out);
    }

    bool loadSeriaFromFile(const std::string &dirPath, const
std::string &fileName, Note *seria, int count) {
        const std::string path = joinPath(dirPath, fileName);
        const std::string content = readFileUtf8(path);
        if (content.empty()) {
            return false;
        }

        const std::vector<std::string> tokens =
splitNoteTokens(content);
        if (static_cast<int>(tokens.size()) != count) {
            return false;
        }

        for (int k = 0; k < count; ++k) {
            if (tokens[k].find(',') != std::string::npos ||
tokens[k].find(';') != std::string::npos) {
                return false;
            }
            if (!parseNoteToken(tokens[k], seria[k])) {
                return false;
            }
        }
        return true;
    }
}

```

Листинг Б.19 – Код «Encoding.h»

```

#pragma once

#include <string>

void initEncodingConsole();

std::string applicationDirectory();

std::wstring utf8ToWide(const std::string &utf8);

std::string wideToUtf8(const std::wstring &wide);

```

Листинг Б.20 – Код «Encoding.cpp»

```

#include "Encoding.h"

#ifdef NOMINMAX
#define NOMINMAX
#endif
#include <windows.h>

#include <string>

```

```

void initEncodingConsole() {
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
}

std::wstring utf8ToWide(const std::string &utf8) {
    if (utf8.empty()) {
        return {};
    }
    const int n = MultiByteToWideChar(CP_UTF8, 0, utf8.data(),
static_cast<int>(utf8.size()), nullptr, 0);
    if (n <= 0) {
        return {};
    }
    std::wstring wide(static_cast<size_t>(n), L'\0');
    MultiByteToWideChar(CP_UTF8, 0, utf8.data(),
static_cast<int>(utf8.size()), wide.data(), n);
    return wide;
}

std::string wideToUtf8(const std::wstring &wide) {
    if (wide.empty()) {
        return {};
    }
    const int n = WideCharToMultiByte(CP_UTF8, 0, wide.data(),
static_cast<int>(wide.size()), nullptr, 0,
                                nullptr, nullptr);
    if (n <= 0) {
        return {};
    }
    std::string utf8(static_cast<size_t>(n), '\0');
    WideCharToMultiByte(CP_UTF8, 0, wide.data(),
static_cast<int>(wide.size()), utf8.data(), n, nullptr,
                                nullptr);
    return utf8;
}

std::string applicationDirectory() {
    wchar_t buf[MAX_PATH] = {0};
    const DWORD n = GetModuleFileNameW(nullptr, buf, MAX_PATH);
    if (n == 0) {
        return ".";
    }
    std::wstring p(buf, buf + n);
    const size_t slash = p.find_last_of(L"\\\/");
    if (slash == std::wstring::npos) {
        return ".";
    }
    return wideToUtf8(p.substr(0, slash));
}

```

Приложение В
(рекомендуемое)
Результаты тестирования

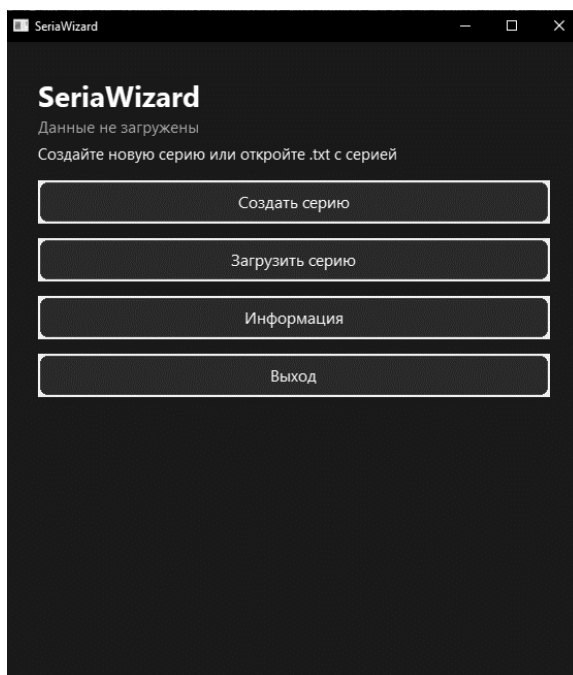


Рисунок В.1 – Результат теста № 1

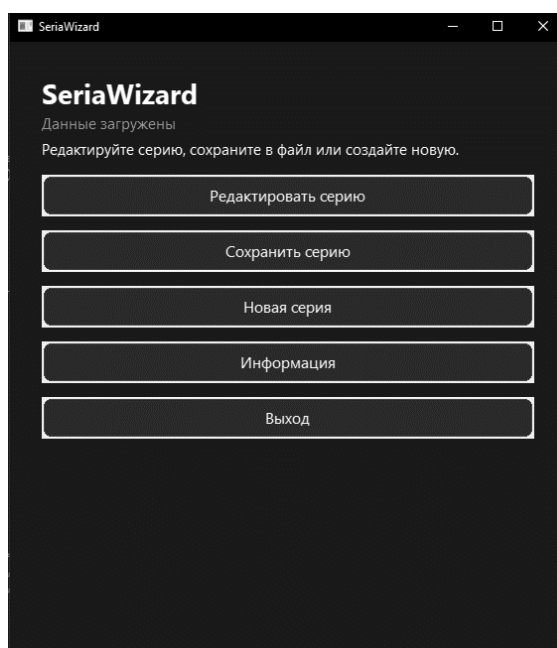


Рисунок В.2 – Результат теста № 2

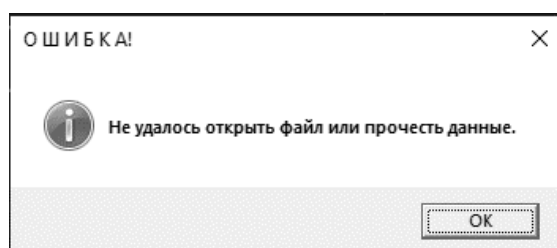


Рисунок В.3 – Результат теста № 3

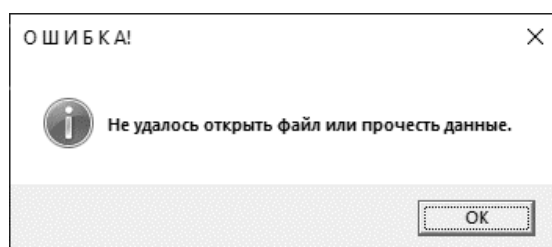


Рисунок В.4 – Результат теста № 4

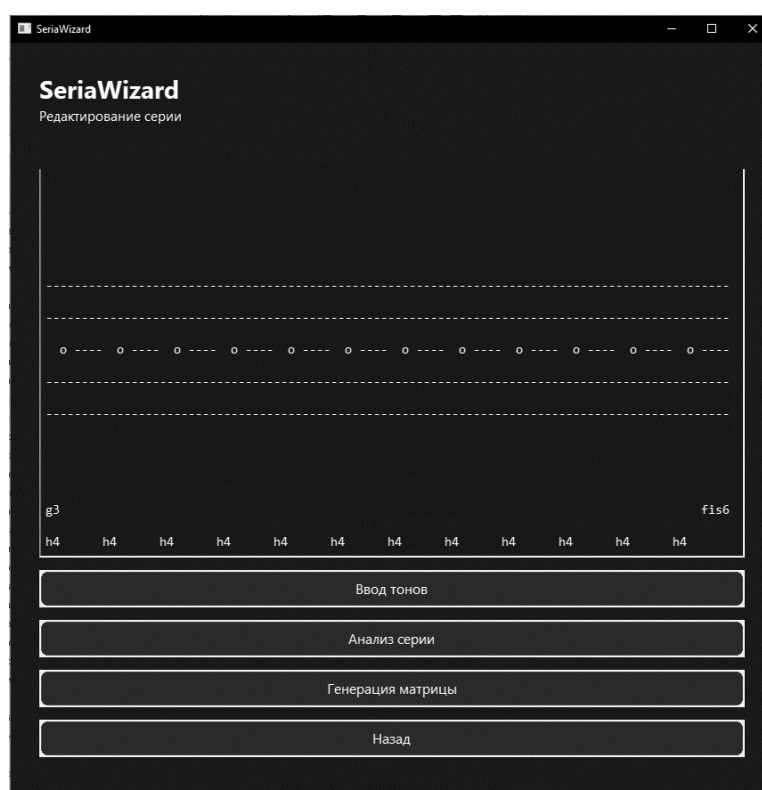


Рисунок В.5 – Результат теста № 5

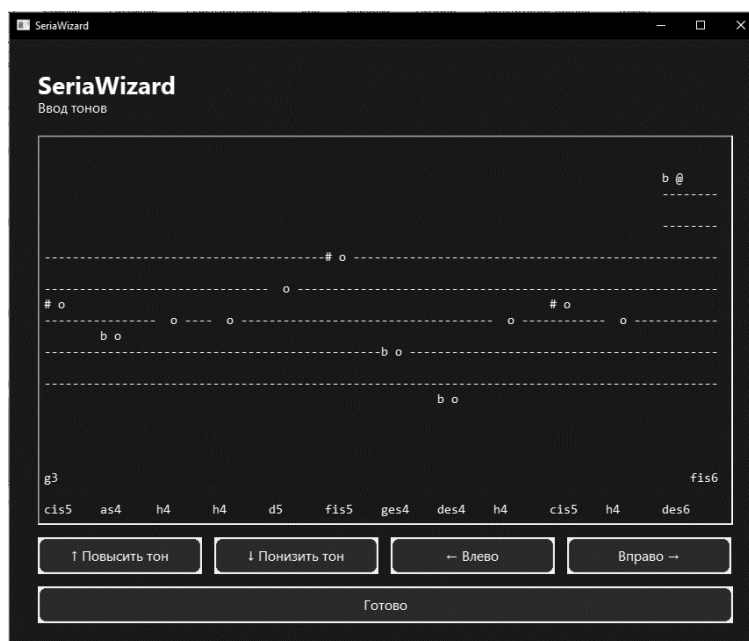


Рисунок В.6 – Результат теста № 6

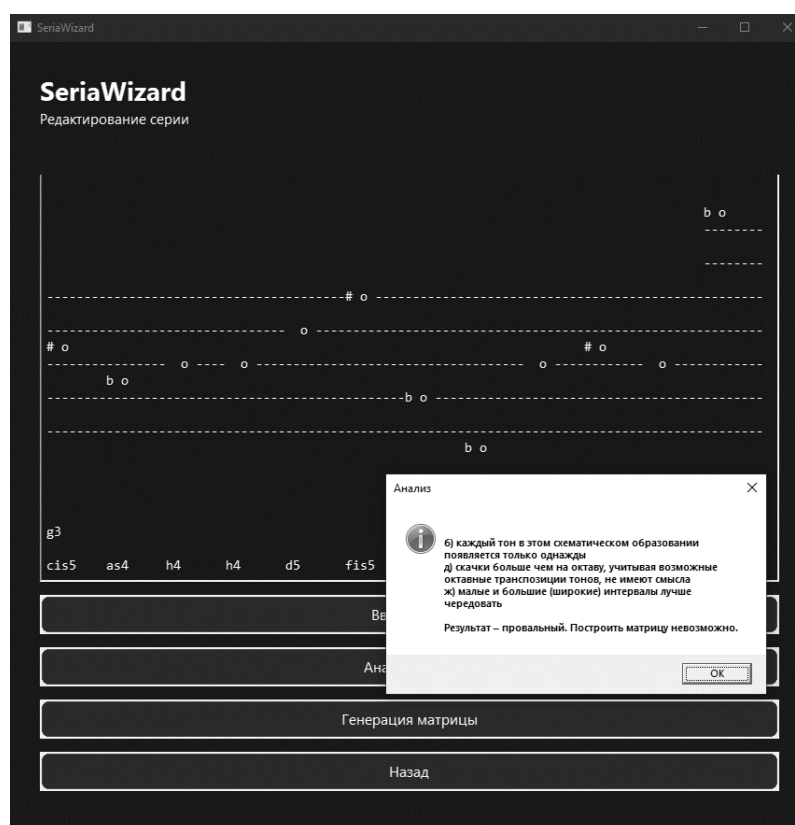


Рисунок В.7 – Результат теста № 7

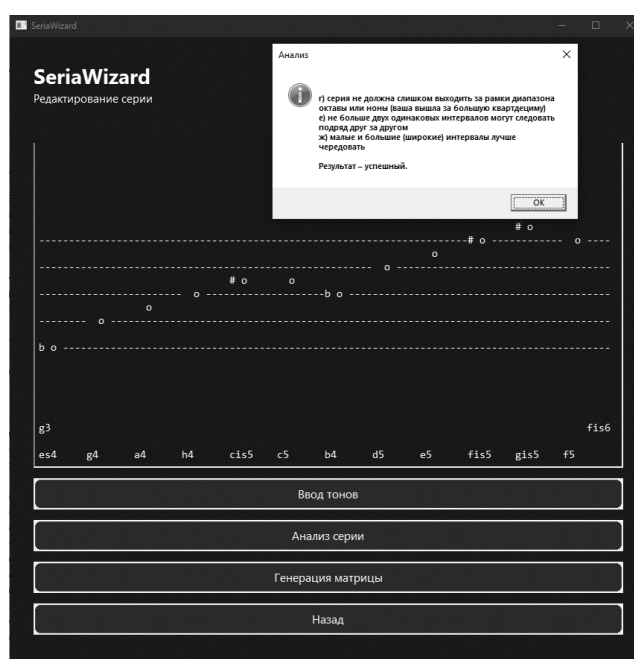


Рисунок В.8 – Результат теста № 8

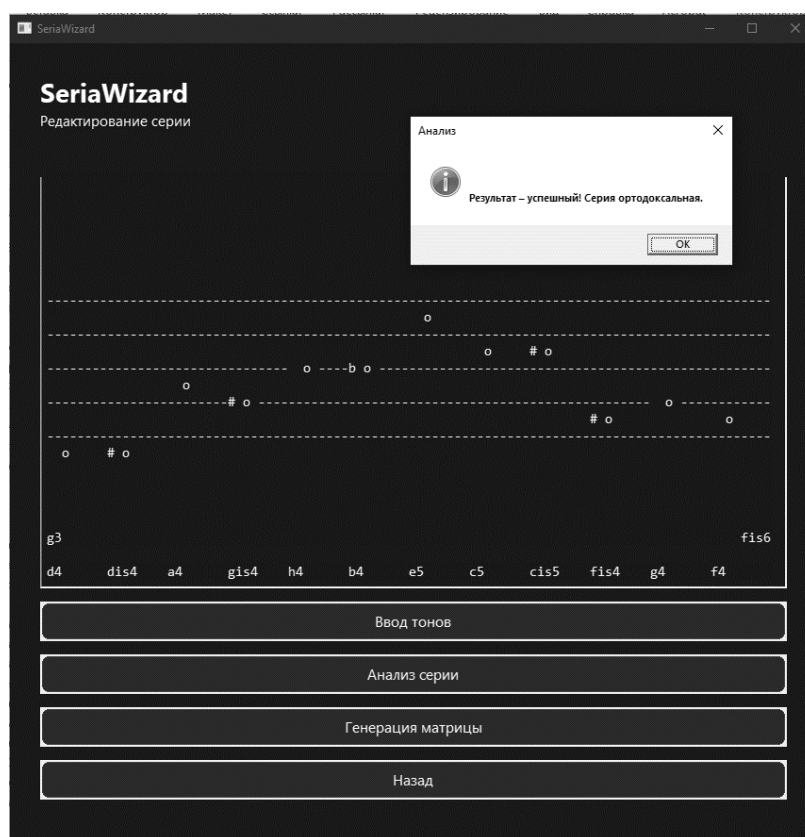


Рисунок В.9 – Результат теста № 9

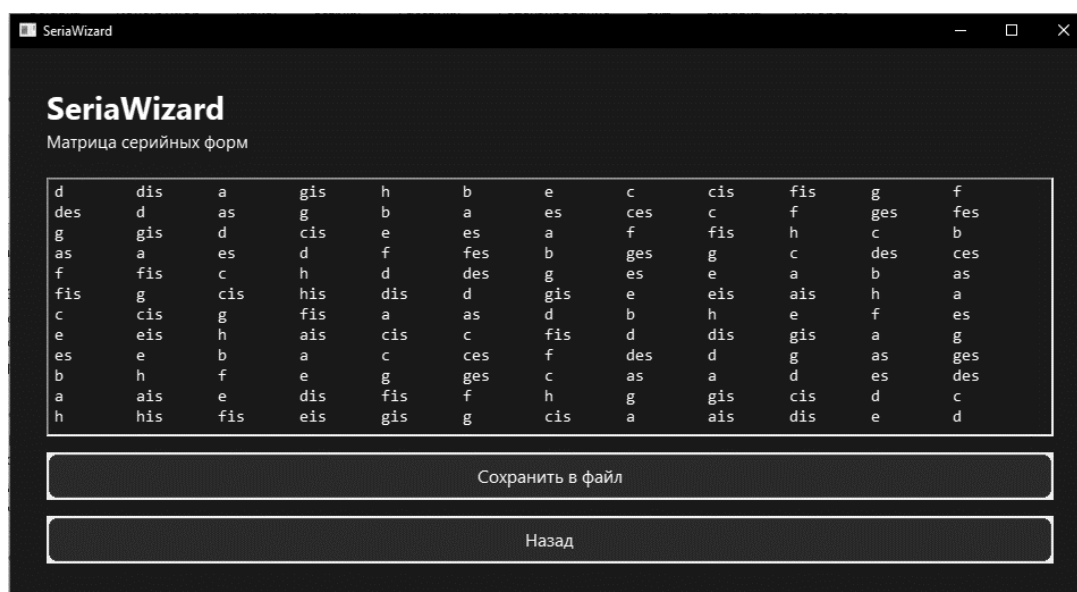


Рисунок В.10 – Результат теста № 10

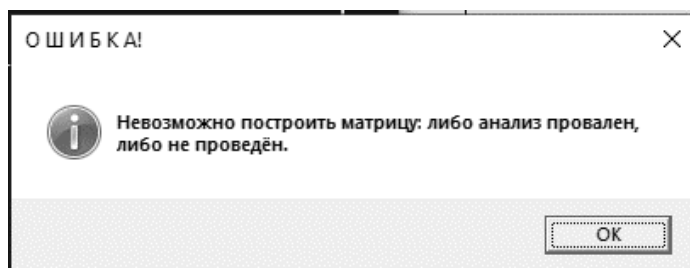


Рисунок В.11 – Результат теста № 11

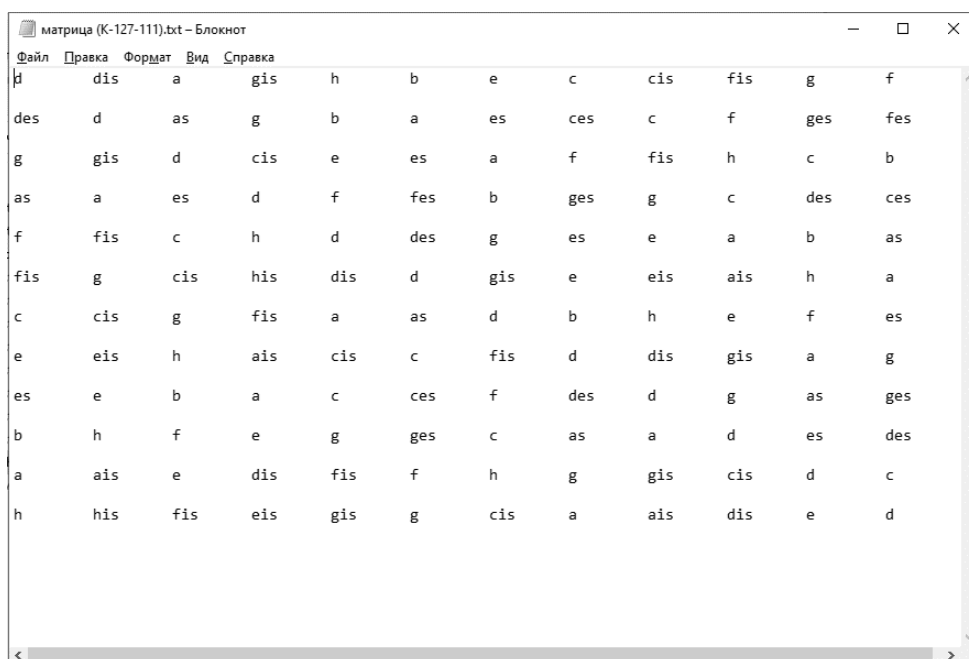


Рисунок В.12 – Результат теста № 12

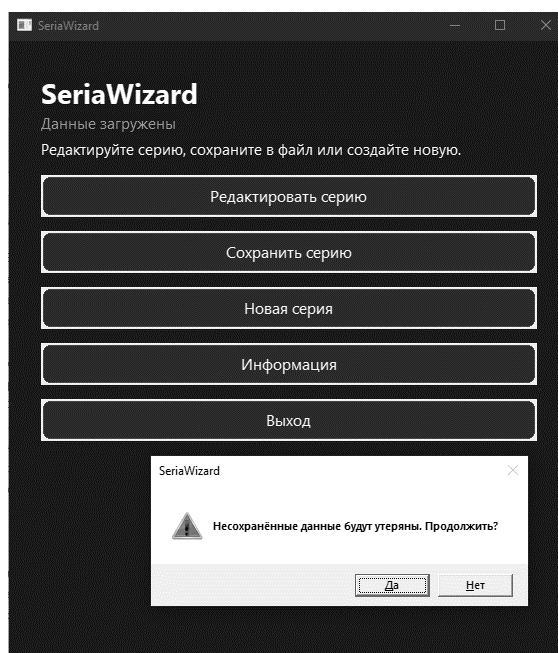


Рисунок В.13 – Результат теста № 13

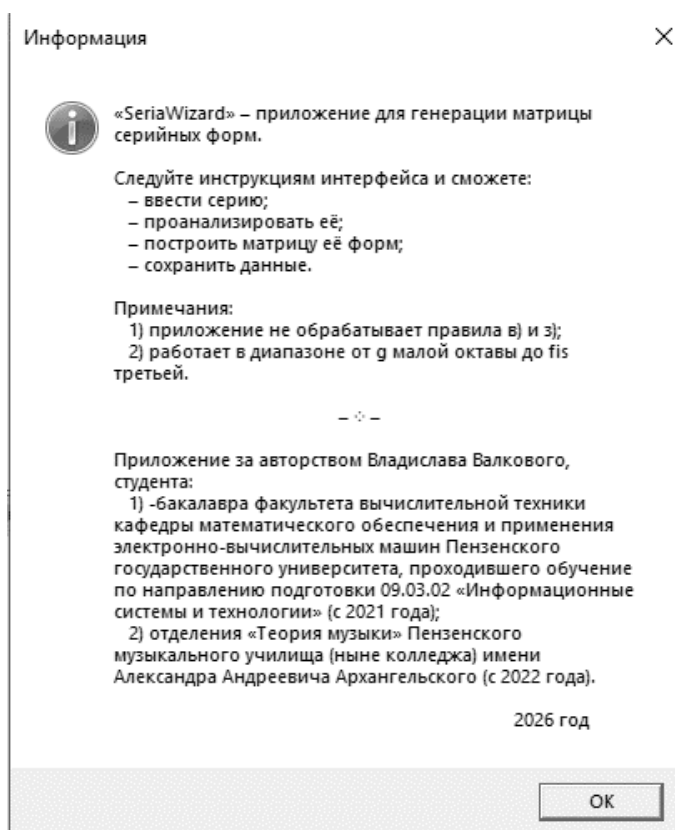


Рисунок В.14 – Результат теста № 14

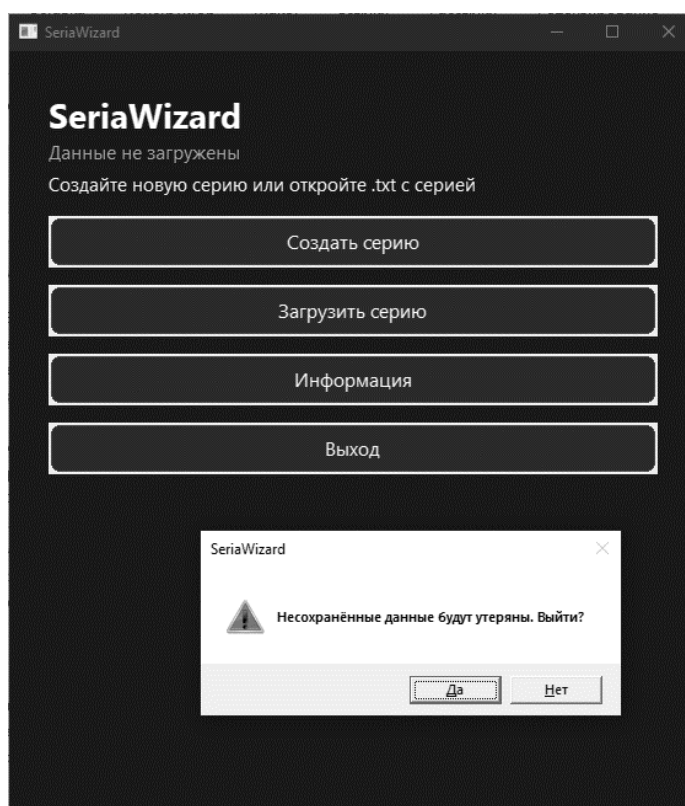


Рисунок В.15 – Результат теста № 15

Приложение Г
(рекомендуемое)
Копия акта о внедрении

АКТ

внедрения приложения для
генерации матрицы серийных форм «SeriaWizard»

Разработанное студентом IV курса Федерального государственного бюджетного образовательного учреждения высшего образования «Пензенский государственный университет», Факультета вычислительной техники, Кафедры математического обеспечения и применения электронно-вычислительных машин Валковым Владиславом Викторовичем приложение для генерации матрицы серийных форм «SeriaWizard» было решено передать в эксплуатацию в Государственное бюджетное профессиональное образовательное учреждение «Пензенский музыкальный колледж имени А. А. Архангельского» для оптимизации работ педагогической и творческой направленностей.

Приложение планируется внедрить 1 сентября 2026 года.

Назначение приложения:

- автоматизация процесса анализа качества данной двенадцатитоновой серии (достаточного для построения матрицы);
- автоматизация процесса построения матрицы форм данной двенадцатитоновой серии.

Эксплуатационные характеристики приложения: справочная документация встроена в приложение в виде контекстной справки.

Для обеспечения работы приложения для генерации матрицы серийных форм «SeriaWizard» необходимо:

- базовая аппаратная конфигурация;
- операционная система Microsoft Windows версии не ниже 7.

Показатели эффективности приложения для генерации матрицы серийных форм «SeriaWizard»:

- сокращение времени анализа серии на 99,9976 %;
- сокращение времени на построение матрицы форм серии на 99,9992 %.

ГБПОУ «Пензенский музыкальный колледж имени А. А. Архангельского» обязуется не передавать разработку Валкова В. В. для использования в сторонних организациях.

Директор:



/ Токарев Э. А. /

«17» 06 2026 г.

Приложение Д
(рекомендуемое)
Диаграмма Ганта

