

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего
образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация
автоматизированных систем специального
назначения»
Специализация: «Эксплуатация вычислительных машин,
комплексов, систем и сетей специального
назначения»

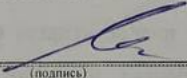
ДИПЛОМНЫЙ ПРОЕКТ

на тему

**РАЗРАБОТКА ПРОГРАММЫ АВТОМАТИЗАЦИИ
ОБРАБОТКИ РЕНТГЕНОВСКИХ СНИМКОВ**

Студент  20.06.2025 Быстров Михаил Сергеевич
(подпись, дата) (ФИО полностью)
Руководитель  23.06.2025 Бычков А.С.
(подпись, дата) (фамилия, инициалы)
Нормоконтролёр  23.06.2025 Бычков А.С.
(подпись, дата) (фамилия, инициалы)
Рецензент кат. Управляющие  Севастьянов А.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2025 2025 г. № 16)

Заведующий кафедрой  Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой удовлетворительно
(протокол заседания ГЭК от 23.06.2025 № 12)

Секретарь ГЭК  Гурин Е.И.
(подпись) (фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
« 06 » 05 2025 г.

**ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ**

1. Студент Быстров Михаил Сергеевич гр. 20ВОЭ1 факультета вычислительной техники.
2. Тема работы: «Разработка программы автоматизации обработки рентгеновских снимков».
Тема утверждена приказом ПГУ № 601/о от 06.05.2025 г.
3. Руководитель работы к.т.н., доцент каф. ВТ «ПГУ» Бычков А.С.
4. Задание на дипломную работу (назначение разработки, исходные данные и т.п.).

Разработать программу автоматизации обработки рентгеновских снимков. Программа должна обеспечивать возможность пользователям загружать цифровые рентгеновские изображения и измерять линейные размеры, диаметры и углы на них с помощью стандартного диалогового окна. На основании анализа этих значений программа должна формировать диагноз заболевания и рекомендации о необходимости или целесообразности проведения той или иной хирургической операции.

При этом необходимо решить задачи создания графического интерфейса, обеспечения загрузки и отображения изображений, разработки интерактивных инструментов для указания точек и построения измеряемых геометрических примитивов, а также вычисление и отображение соответствующих значений.

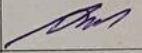
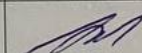
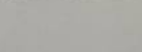
Программу реализовать на языке программирования C# с использованием технологии Windows Forms.

5. Перечень подлежащих разработке вопросов.

Требуется провести анализ технического задания на проектирование, изучить словесное описание последовательности действий медицинского персонала при анализе параметров

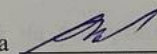
рентгеновских снимков, разработать алгоритмы работы программы автоматизации обработки рентгеновских снимков, расчета и выдачи пациентам соответствующих вердиктов, выполнить реализацию программы на языке C# и ее тестирование.

6. Календарный график выполнения работы.

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области	10	06.05.2025 – 08.05.2025	
2	Проектирование программы автоматизации обработки рентгеновских снимков	10	09.05.2025 – 11.05.2025	
3	Разработка программы автоматизации обработки рентгеновских снимков	30	12.05.2025 – 19.05.2025	
4	Отладка и тестирование	10	20.05.2025 – 22.05.2025	
5	Разработка экономической части	5	23.05.2025	
6	Разработка раздела БЖД	5	24.05.2025	
7	Оформление графической части	10	25.05.2025 – 27.05.2025	
8	Оформление выпускной работы	20	28.05.2025 – 30.05.2025	

Дата выдачи задания « 06 » мая 2025 г.

Руководитель дипломного проекта


(подпись, дата)

06.05.2025 Бычков А.С.

(фамилия, инициалы)

Задание к исполнению принял студент


(подпись, дата)

06.05.2025 Быстров М.С.

(фамилия, инициалы)

Работу к защите допустить
Декан факультета


(подпись, дата)

24.06.2025 Фионова Л.Р.

(фамилия, инициалы)

Реферат

Пояснительная записка содержит 158 листов, 36 рисунков, 6 таблиц, 8 источников и 3 приложения.

ИЗМЕРЕНИЕ ПАРАМЕТРОВ РЕНТГЕНОВСКИХ СНИМКОВ, ТАЗОБЕДРЕННЫЙ СУСТАВ, АВТОМАТИЗАЦИЯ ОБРАБОТКИ РЕНТГЕНОВСКИХ СНИМКОВ, ЭКРАННАЯ ФОРМА, ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

Целью данной работы является создание программы автоматизации обработки рентгеновских снимков и реализация ее на языке программирования C# с использованием технологии Windows Forms.

Для достижения цели необходимо решить задачи создания графического интерфейса, обеспечения загрузки и отображения изображений, разработки интерактивных инструментов для указания точек и построения измеряемых геометрических примитивов, а также вычисления и визуализации соответствующих значений.

Объектом исследования является процесс количественного анализа информации, отображаемой на рентгеновских снимках, результатом которого является формирование диагноза заболевания и методов его лечения.

Предметом исследования является разработка программного обеспечения, его архитектуры, алгоритмов обработки пользовательского ввода и визуализации результатов измерений.

Так как разработанная программа, помимо измерения и анализа заданных параметров, позволяет определить диагноз заболевания пациента и сформировать рекомендации по его лечению, то тематика работы является, безусловно, важной и актуальной и проект может быть рекомендован к внедрению.

					ПГУ1.090501.04.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата				
Разраб.		Быстров М.А.		20.06.25	Программа автоматизации обработки рентгеновских снимков	Лит.	Лист	Листов
Провер.		Бычков А.С.		23.06.25			2	158
Н. Контр		Бычков А.С.		23.06.25		ФВТ, каф. ВТ гр. 20ВОЭ1		
Утверд.		Митрохин М.А.		23.06.25	Пояснительная записка			

Содержание

Перечень сокращений и обозначений	5
Введение	6
1 Разработка программы автоматизации обработки рентгеновских снимков	8
1.1 Техническое задание на проектирование	8
1.2 Разрботка алгоритма работы программы	9
1.2.1 Разработка процедуры ввода исходных данных о пациенте, измерения и расчета параметров рентгенограммы	10
1.2.2 Разработка процедуры обработки параметров рентгенограмм	14
1.3 Разработка экранных форм для ввода параметров рентгенограмм	27
1.3.1 Краткое описание экранных форм	27
2 Реализация программы автоматизации обработки рентгеновских снимков на языке высокого уровня	39
2.1 Выбор языка программирования и среды разработки	39
2.2 Описание программы автоматизации обработки рентгеновских снимков .	40
2.2.1 Общая структура программы	40
2.2.2 Описание файлов и их назначение	40
2.2.2.1 Файл Program.cs	40
2.2.2.2 Файл SharedData.cs	41
2.2.2.3 Файлы Window_(1-9).Designer.cs	42
2.2.2.4 Файлы Window_(1-9).cs	42
2.2.2.5 Повторяющиеся элементы интерфейса и логика их работы	42
2.2.2.6 Рисование линий	49
2.2.2.7 Рисование окружности	52
2.2.2.8 Измерение угла	54
3 Руководство пользователя	59
3.1 Краткое описание функций программы	59
3.2 Подробное описание функций программы	59
3.2.1 Открытие рентгенограммы	59

3.2.2	Получение текстовых подсказок	59
3.2.3	Измерение длин	60
3.2.4	Измерение углов	60
3.2.5	Переход к «Шпаргалке»	60
3.2.6	Кнопка «Рассчитать»	61
3.2.7	Кнопка «Далее»	61
3.2.8	Ввод данных пациента	61
3.2.9	Формирование вердикта	61
4	Тестирование программы	63
5	Расчет расходов на создание и внедрение программного обеспечения	77
5.1	Организация процесса разработки программы для автоматизации обработки рентгеновских снимков	77
5.2	Определение себестоимости разработки программы для автоматизации обработки рентгеновских снимков	79
6	Безопасность жизнедеятельности. Организация рабочего места с ПК	81
6.1	Требования к помещению для работы с ПК	81
6.2	Требования к организации рабочих мест	84
	Заключение	87
	Список использованных источников	88
	Приложение А – Схемы алгоритмов	89
	Приложение Б – Листинг программы	98
	Приложение В – Презентация	143

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

АК - ацетабулярный коэффициент

ОДА – опорно-двигательный аппарат

ПК – персональный компьютер

ТБС – тазобедренный сустав

ТЗ – техническое задание на проектирование

C# – современный, с открытым исходным кодом, кросс-платформенный объектно-ориентированный язык программирования

Da - диаметр ацетабулярного фрагмента бедра

Dh - диаметр головки бедра

ICAS - индекс конгруэнтности или соответствия

ISA - индекс сферичности ацетабулярного фрагмента

ISh - индекс сферичности головки бедра

MAUI (.NET Multi-platform App UI) - кроссплатформенный фреймворк для создания нативных мобильных и десктопных приложений с использованием языка программирования C# и языка разметки XAML

.NET — платформа для разработчиков программного обеспечения, состоящая из инструментов, языков программирования и библиотек для создания различных типов приложений

R - расстояние между фигурами слезы

Visual Studio — полнофункциональная интегрированная среда разработки для написания, отладки, тестирования и развертывания кода на различных языках, в том числе и на C#

Windows Forms - платформа разработки приложений. Поддерживает широкий набор функций для разработки, включая элементы управления, графику, привязку данных и ввод пользователя

WPF (Windows Presentation Foundation) — технологии реализации пользовательских интерфейсов для Windows-приложений

					ПГУ1.090501.04.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Введение

Важной частью современной медицины являются высокотехнологичные методы визуализации, среди которых рентгенография занимает одно из ключевых мест благодаря своей доступности и информативности. Точное интерпретирование рентгеновских изображений, которое включает измерение размеров различных анатомических структур и патологических изменений, имеет решающее значение для постановки диагноза, планирования лечения и мониторинга его эффективности. В повседневной практике врачи часто вынуждены измерять расстояния между контрольными точками, диаметры образований или углы, описывающие положение костных структур. Автоматизация этого процесса с использованием специализированного программного обеспечения может не только повысить точность измерений, но и значительно сократить время, затрачиваемое специалистом.

Актуальность данного проекта обусловлена необходимостью разработки доступного и удобного инструмента для проведения анализа рентгеновских снимков и вынесения вердиктов. Целью данной работы является создание программы автоматизации обработки рентгеновских снимков на языке программирования C# с использованием технологии Windows Forms. Программа позволит пользователям загружать цифровые рентгеновские изображения и измерять линейные размеры, диаметры и углы на них с помощью стандартного диалогового окна. Для достижения цели необходимо решить задачи создания графического интерфейса, обеспечения загрузки и отображения изображений, разработки интерактивных инструментов для указания точек и построения измеряемых геометрических примитивов, а также вычисления и отображения соответствующих значений.

Объектом исследования является процесс количественного анализа медицинских рентгеновских снимков. Предметом исследования является

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

разработанное программное обеспечение, его архитектура, алгоритмы обработки пользовательского ввода и визуализации результатов измерений.

1 Разработка программы автоматизации обработки рентгеновских снимков

1.1 Техническое задание на проектирование

Заказчиками разработки программы автоматизации обработки рентгеновских снимков являются представители Национального медицинского исследовательского центра травматологии и ортопедии им. академика Г.А. Илизарова, г. Курган. В качестве ТЗ на проектирование ими были представлены подробные словесные описания алгоритмов действий пользователей программы (хирургов, травматологов, ортопедов) при получении информации о заболеваниях опорно-двигательного аппарата человека, в частности, тазобедренного сустава [1], путем анализа рентгенограмм таза, полученных в разных проекциях. Всего анализу подлежат четыре различные рентгенограммы, три из которых являются обязательными, а четвертая требуется в лишь в определенных исключительных случаях. Необходимость ее применения выясняется в ходе анализа данных первых трех рентгенограмм.

Анализ рентгенограмм включает измерение геометрических параметров суставов, на основании которых делается расчет ряда коэффициентов, характеризующих то или иное заболевание пациента. На основании этого анализа врач выносит вердикт, в котором указывается диагноз заболевания и рекомендации о необходимости или целесообразности проведения той или иной хирургической операции.

Всего в ТЗ приведено словесное описание трех различных алгоритмов действий пользователей для разных возрастных групп пациентов, так как в зависимости от возраста применяются несколько различные подходы при определении диагноза пациента и выработки рекомендаций по его лечению:

- возраст 13 лет и более;
- возраст от 10 до 13 лет;

					ПГУ1.090501.04.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

- возраст от 7 до 9 лет.

1.2 Разработка алгоритма работы программы

В результате изучения словесного описания последовательности действий врачей-хирургов при анализе параметров рентгенограмм, были разработаны алгоритмы работы программы автоматизации обработки рентгеновских снимков, расчета и выдачи пациентам соответствующих вердиктов.

Анализ ТЗ показал, что последовательности действий врачей-хирургов при работе с пациентами различных возрастных групп во многом идентичны друг другу. Особенно это касается процессов измерения геометрических параметров суставов по рентгенограммам, тогда как расчетные формулы часто различны и определяются возрастом пациента. Поэтому на первом этапе разработки алгоритма работы программы была произведена минимизация словесного описания алгоритма путем исключения одинаковых фрагментов для разных возрастных групп.

В результате этого в структуре разрабатываемой программы укрупненно можно выделить две основные части (модуля):

- первый модуль - процедура ввода исходных данных о пациенте и параметров рентгенограмм. Этот модуль используется в любом случае, независимо от возраста пациента;

- второй модуль - процедура обработки введенной информации, формирования вердикта и выдачи его на печать (в текстовый файл для дальнейшей распечатки) или на экран. Данный модуль состоит из трех функционально законченных фрагментов программы (функций) для соответствующих возрастных групп, так как в зависимости от возраста применяются несколько различные подходы при определении диагноза пациента и выработки рекомендаций по его лечению.

Для пациентов в возрасте до 7 лет данная методика диагностики и лечения, а соответственно данная программа неприменимы.

					ПГУ1.090501.04.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

1.2.1 Разработка процедуры ввода исходных данных о пациенте, измерения и расчета параметров рентгенограммы

Первый модуль, процедура ввода исходных данных о пациенте, измерения и расчета параметров рентгенограммы, применяется для всех возрастных групп.

Схема алгоритма первого модуля приведена на рисунке 1 (а - г). На схеме наглядно представлена последовательность операций, выполняемых пользователем при вводе исходных данных с рентгеновских снимков.

На выходе данной процедуры, помимо информации о фамилии, имени, отчестве и возрасте пациента, имеем ряд переменных, содержащих результаты измерений и расчета требуемых параметров, необходимых для вынесения вердикта:

- Dh (диаметр головки бедра);
- Da (диаметр ацетабулярного фрагмента бедра);
- R (расстояние между фигурами слезы);
- ISh (индекс сферичности головки бедра);
- Dist_A, Dist_B, Dist_D, Dist_W (различные измеряемые расстояния);
- Okano (индекс Окано);
- Angle_A, Angle_B, Angle_V, Angle_E (измеряемые углы);
- ISA (индекс сферичности ацетабулярного фрагмента);
- ICAS (индекс конгруэнтности или соответствия);
- AK (ацетабулярный коэффициент);
- UOB, SLN, SPN_1, SDLN, HP, BP, trans (дополнительные параметры);
- PHIO (ФИО);
- year (год рождения пациента);
- age (возраст пациента, количество полных лет).

Все вышеперечисленные переменные являются исходными данными для второго модуля – модуля обработки введенной информации, формирования и выдачи вердикта.

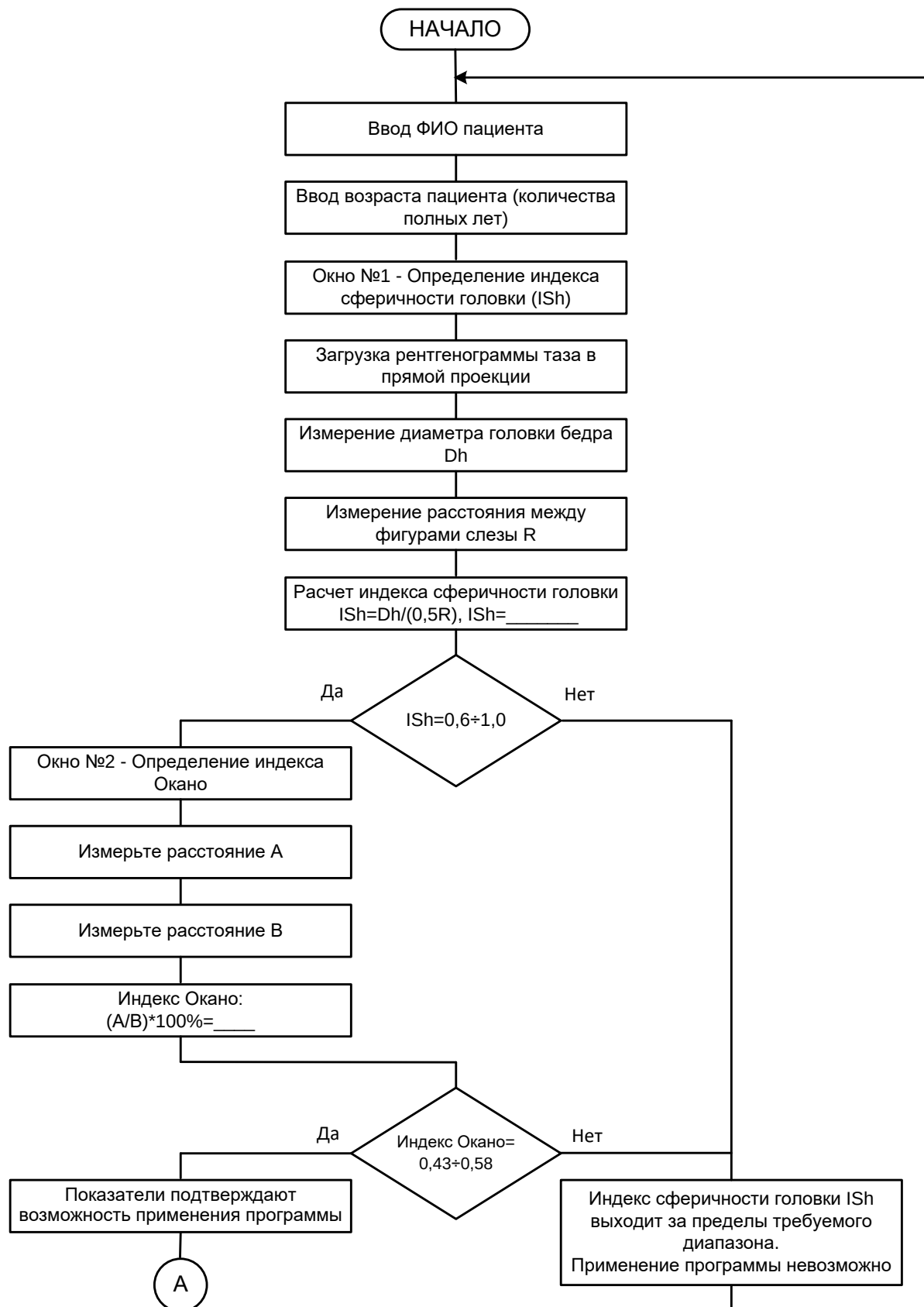


Рисунок 1 (а) – Схема алгоритма ввода исходных данных о пациенте и параметров рентгенограмм

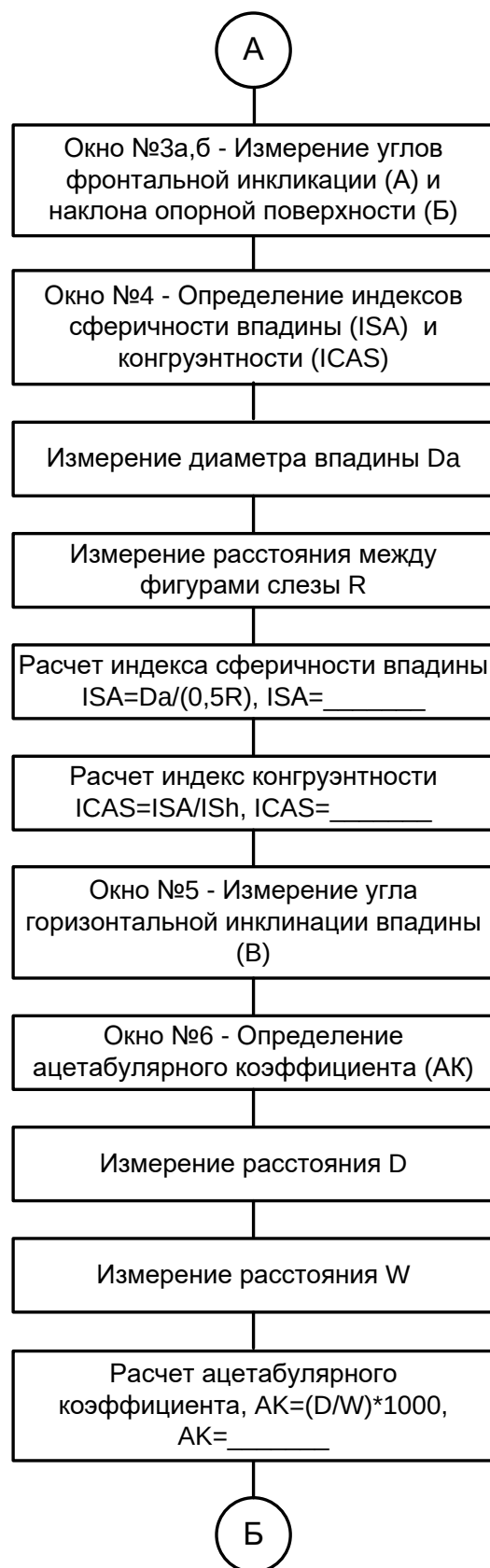


Рисунок 1 (б) – Схема алгоритма ввода исходных данных о пациенте и параметров рентгенограмм (продолжение)

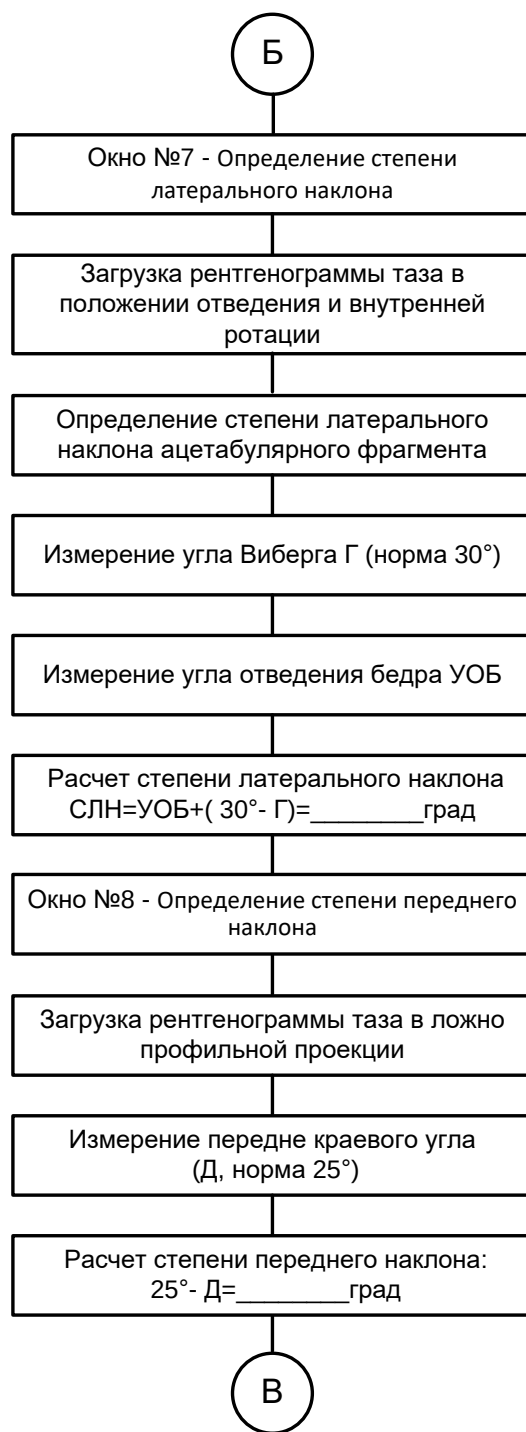


Рисунок 1 (в) – Схема алгоритма ввода исходных данных о пациенте и параметров рентгенограмм (продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

13

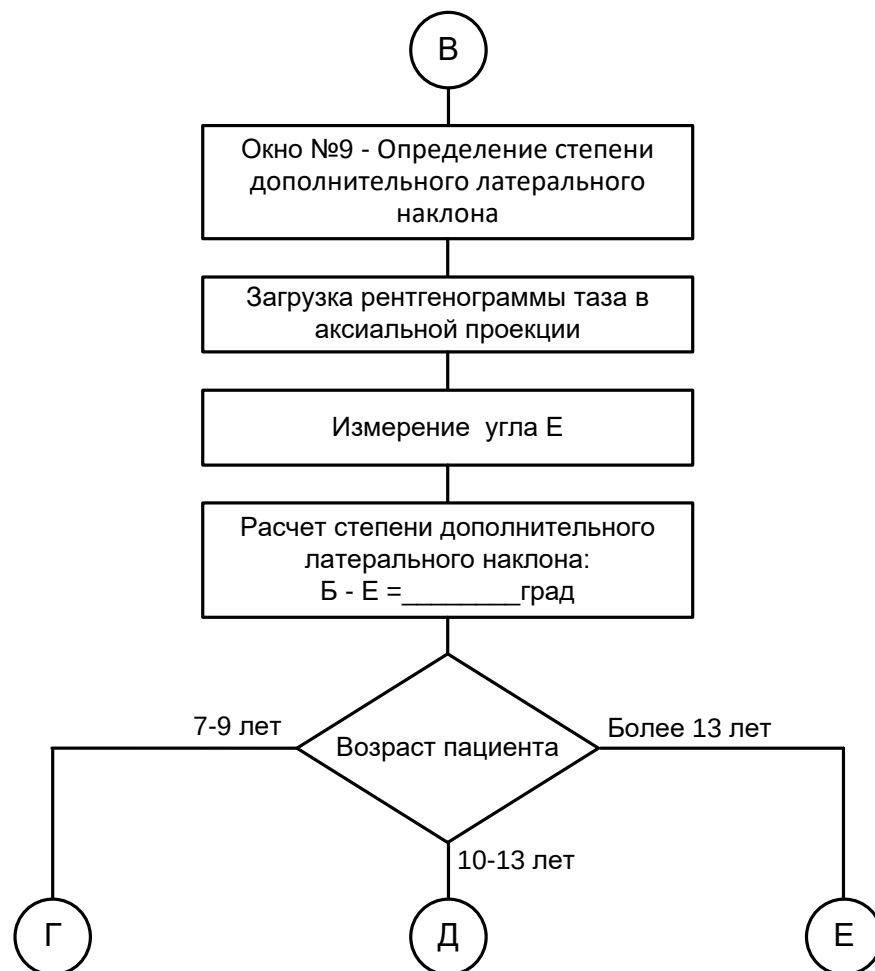


Рисунок 1 (г) – Схема алгоритма ввода исходных данных о пациенте и параметров рентгенограмм (продолжение)

1.2.2 Разработка процедуры обработки параметров рентгенограмм

По завершению процедуры ввода исходных данных о пациенте в зависимости от его возраста выполняется соответствующая функция обработки введенной информации, формирования и выдачи вердикта. Схемы алгоритмов для разных возрастных групп приведены на рисунках, представленных ниже.

Начнем изложение общего алгоритма анализа рентгенограмм и вынесения соответствующего вердикта с рассмотрения алгоритма для наиболее многочисленной группы пациентов, возраст которых составляет 13 лет и более. Схема алгоритма обработки параметров рентгенограмм приведена ниже, на рисунке 2 (а - г).

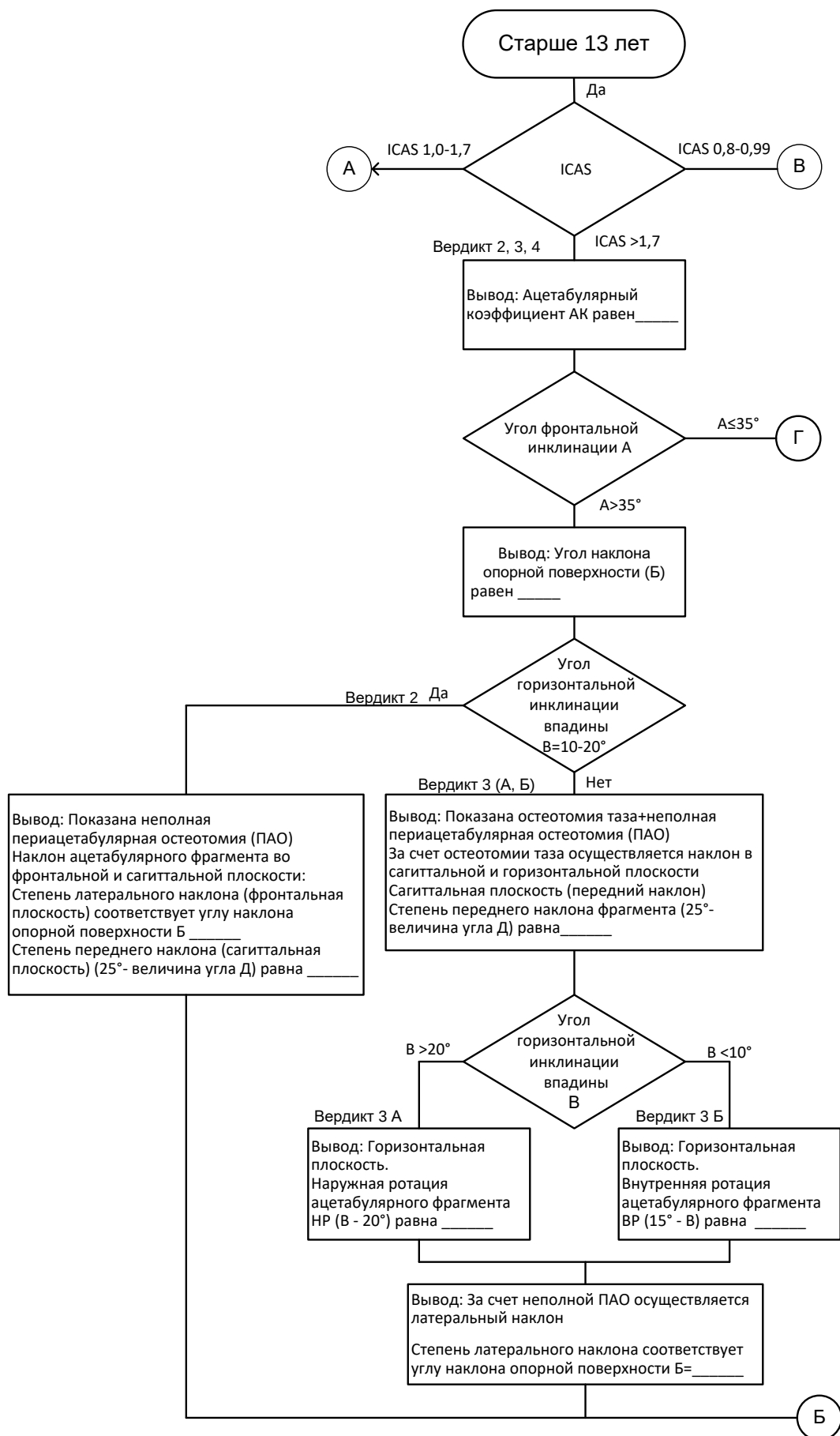


Рисунок 2 (а) – Схема алгоритма обработки параметров рентгенограмм

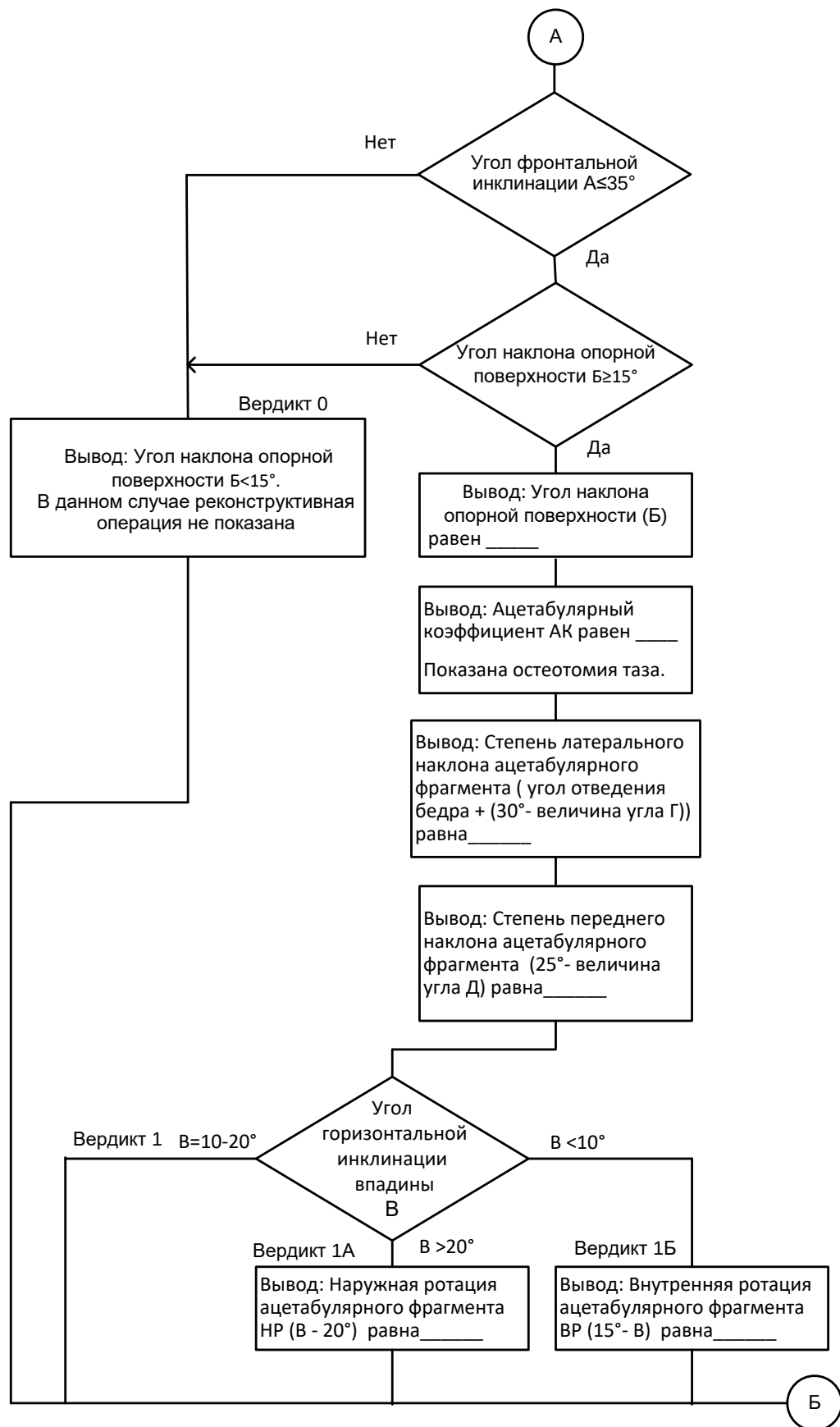


Рисунок 2 (б) – Схема алгоритма обработки параметров рентгенограмм
(продолжение)

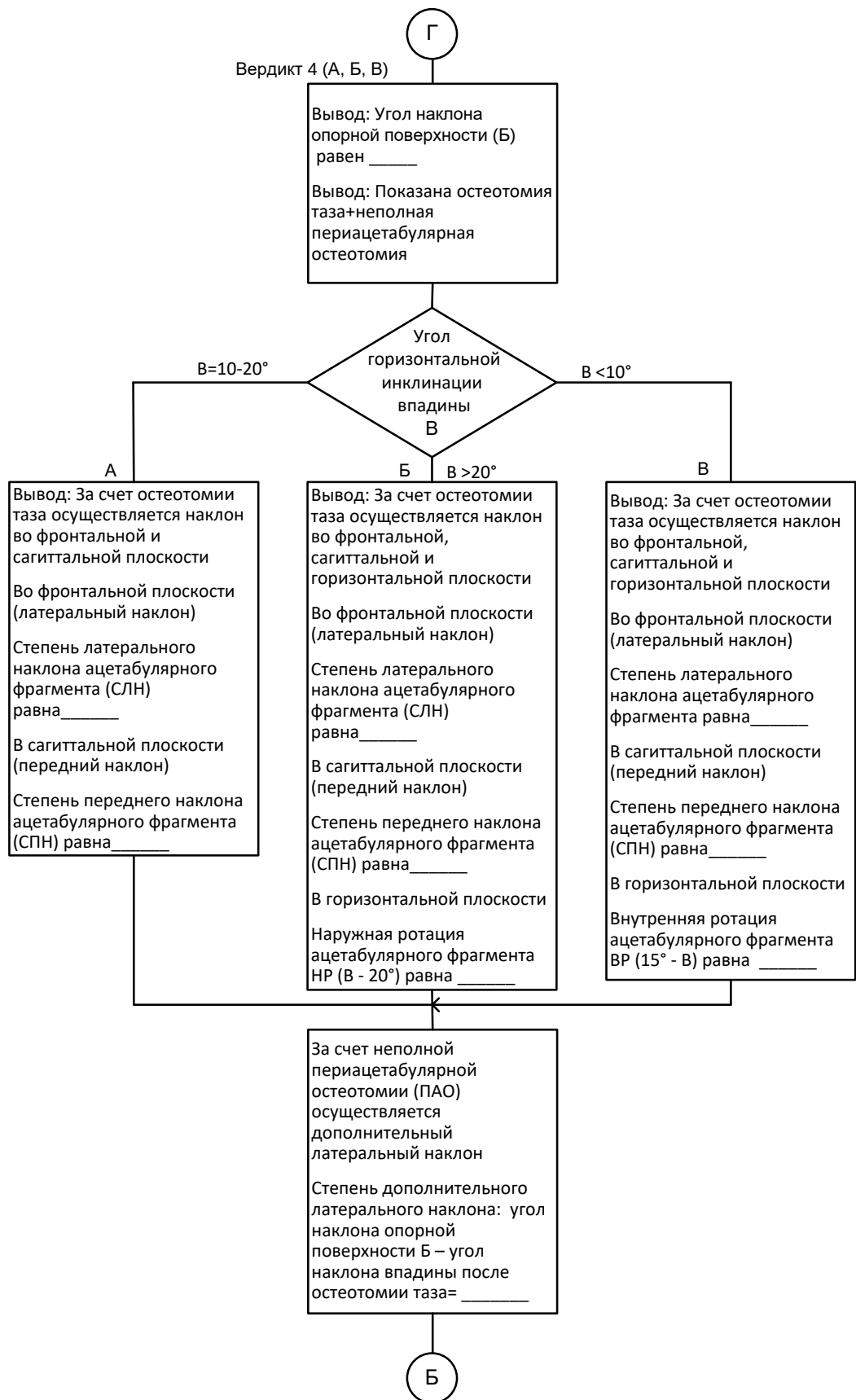


Рисунок 2 (в) – Схема алгоритма обработки параметров рентгенограмм
(продолжение)

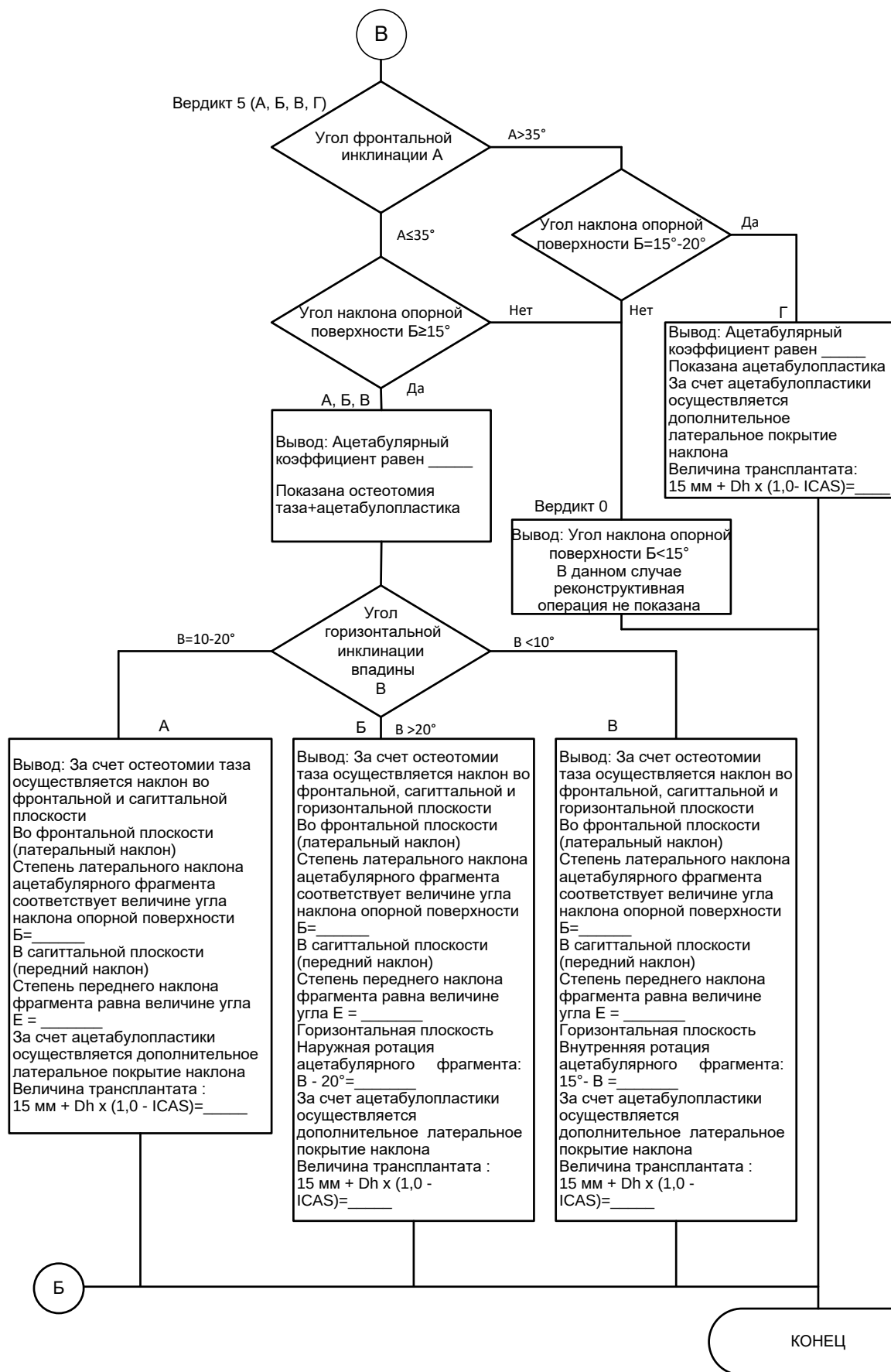


Рисунок 2 (г) – Схема алгоритма обработки параметров рентгенограмм
(продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

18

Схема алгоритма обработки параметров рентгенограмм для возрастной группы от 10 до 13 приведена ниже, на рисунке 3 (а - г).

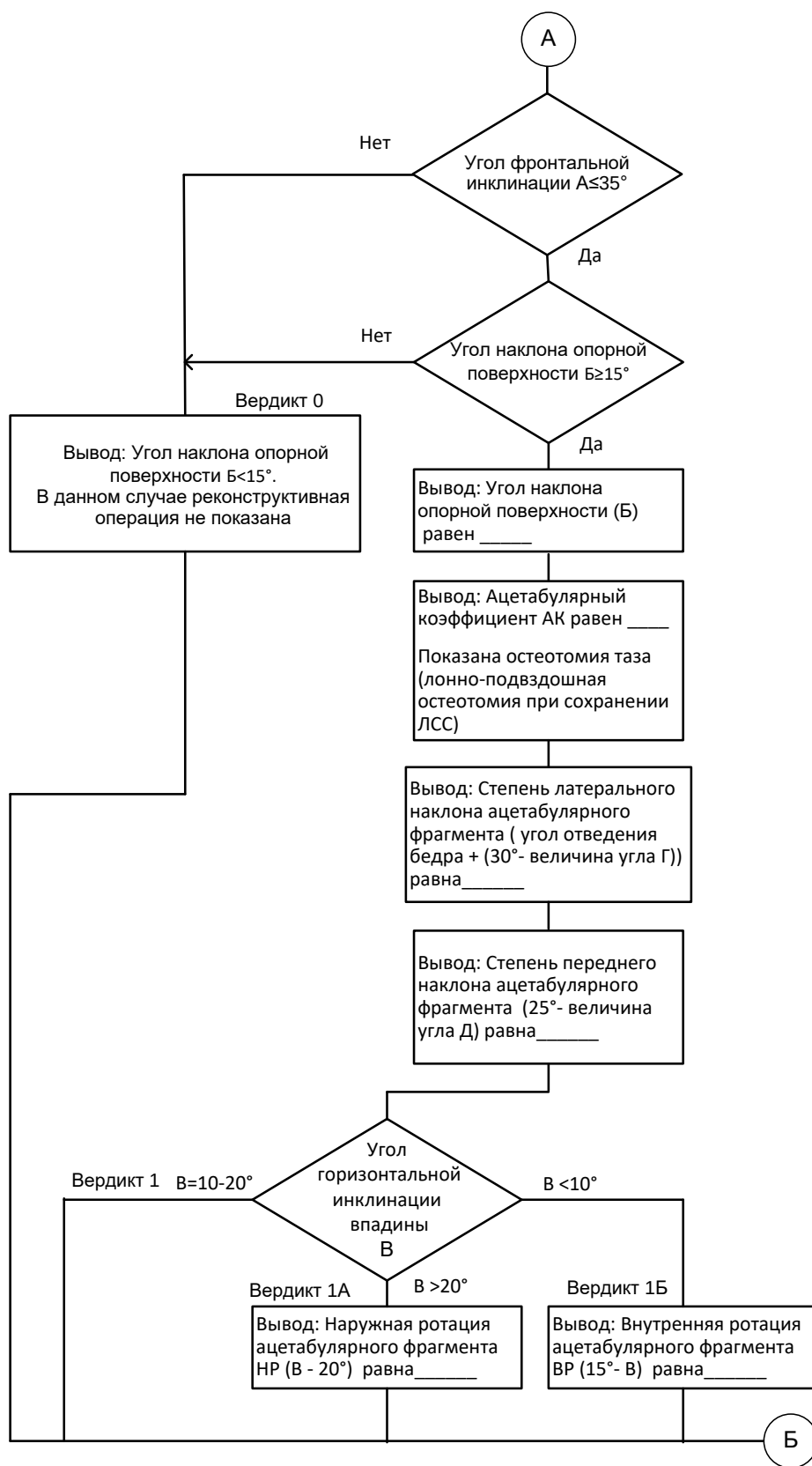


Рисунок 3 (а) – Возрастная группа от 10 до 13 лет. Схема алгоритма

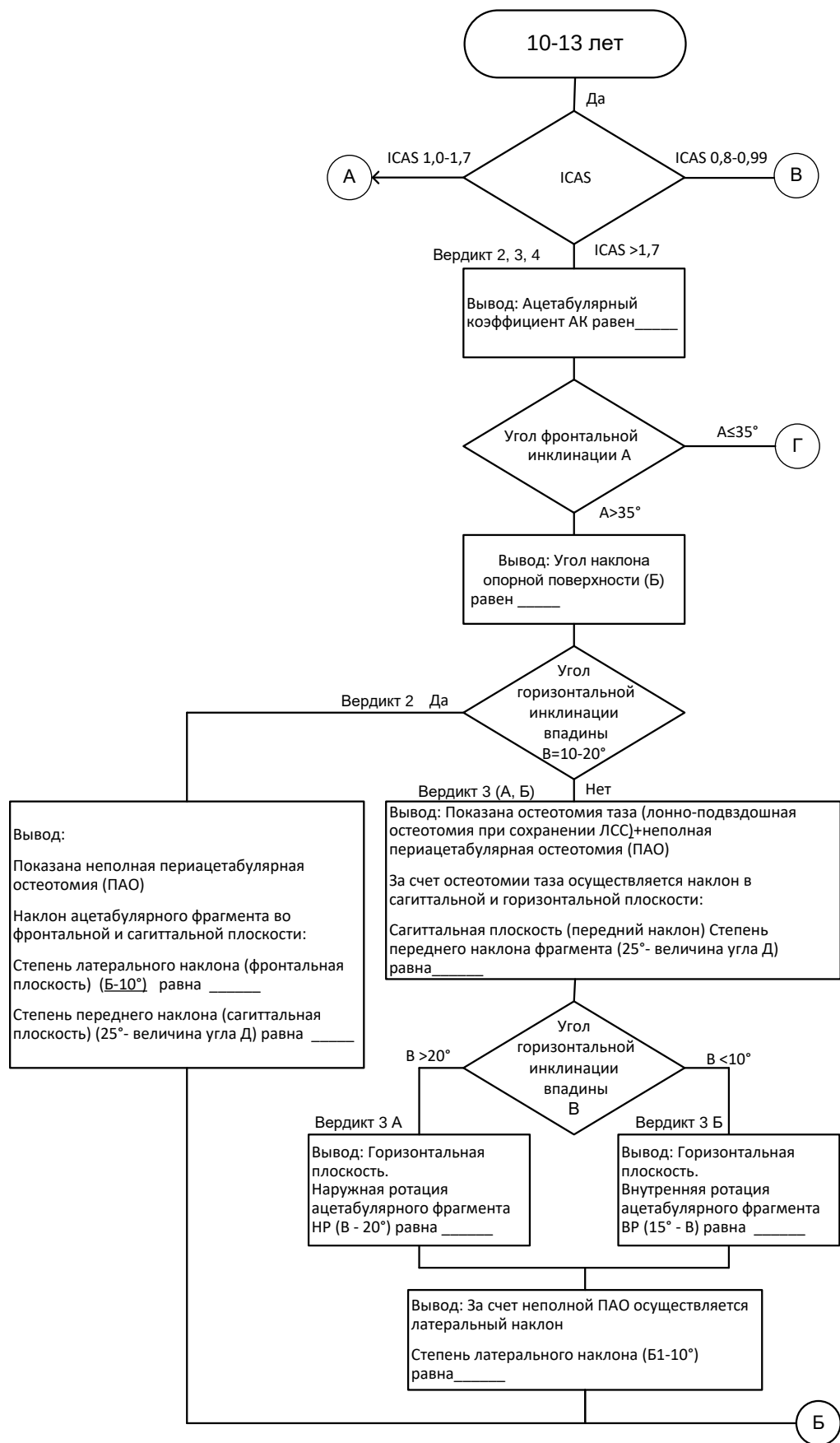


Рисунок 3 (б) – Возрастная группа от 10 до 13 лет.

Схема алгоритма (продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

20



Рисунок 3 (в) – Возрастная группа от 10 до 13 лет.

Схема алгоритма (продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

21

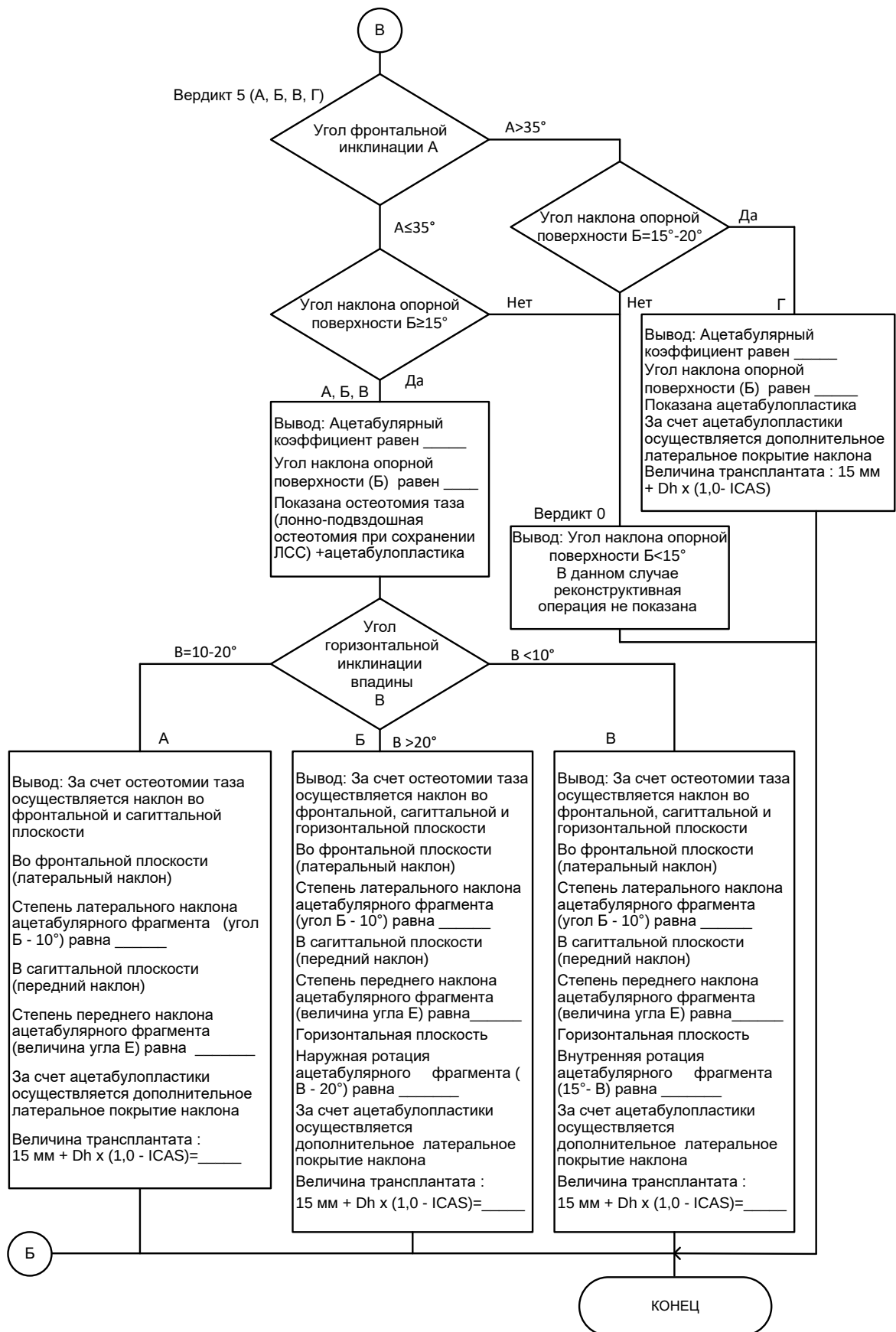


Рисунок 3 (г) – Возрастная группа от 10 до 13 лет.

Схема алгоритма (продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

22

Схема алгоритма обработки параметров рентгенограмм для возрастной группы от 7 до 9 приведена ниже, на рисунке 4 (а - г).

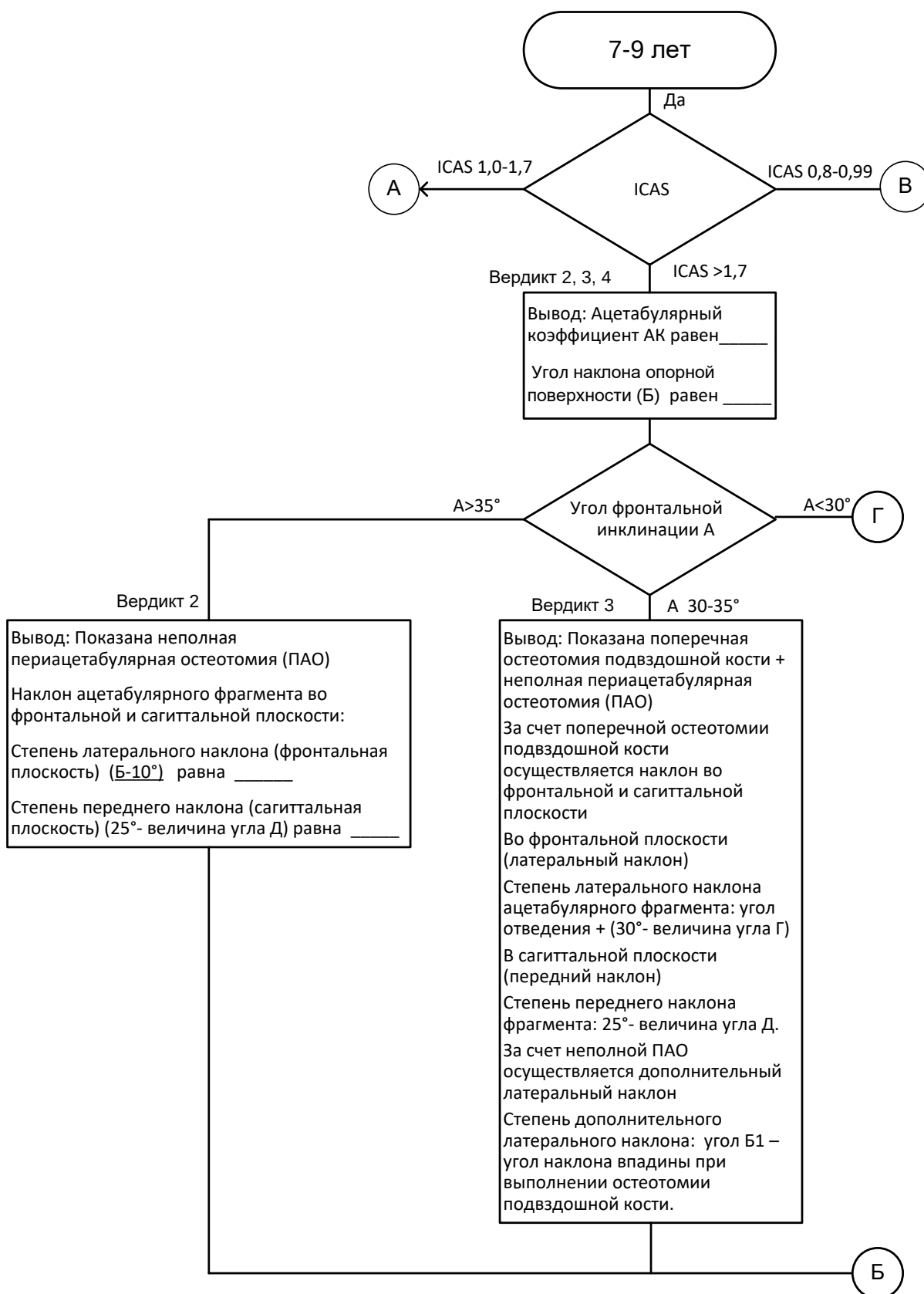


Рисунок 4 (а) – Возрастная группа от 7 до 9 лет. Схема алгоритма

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

23

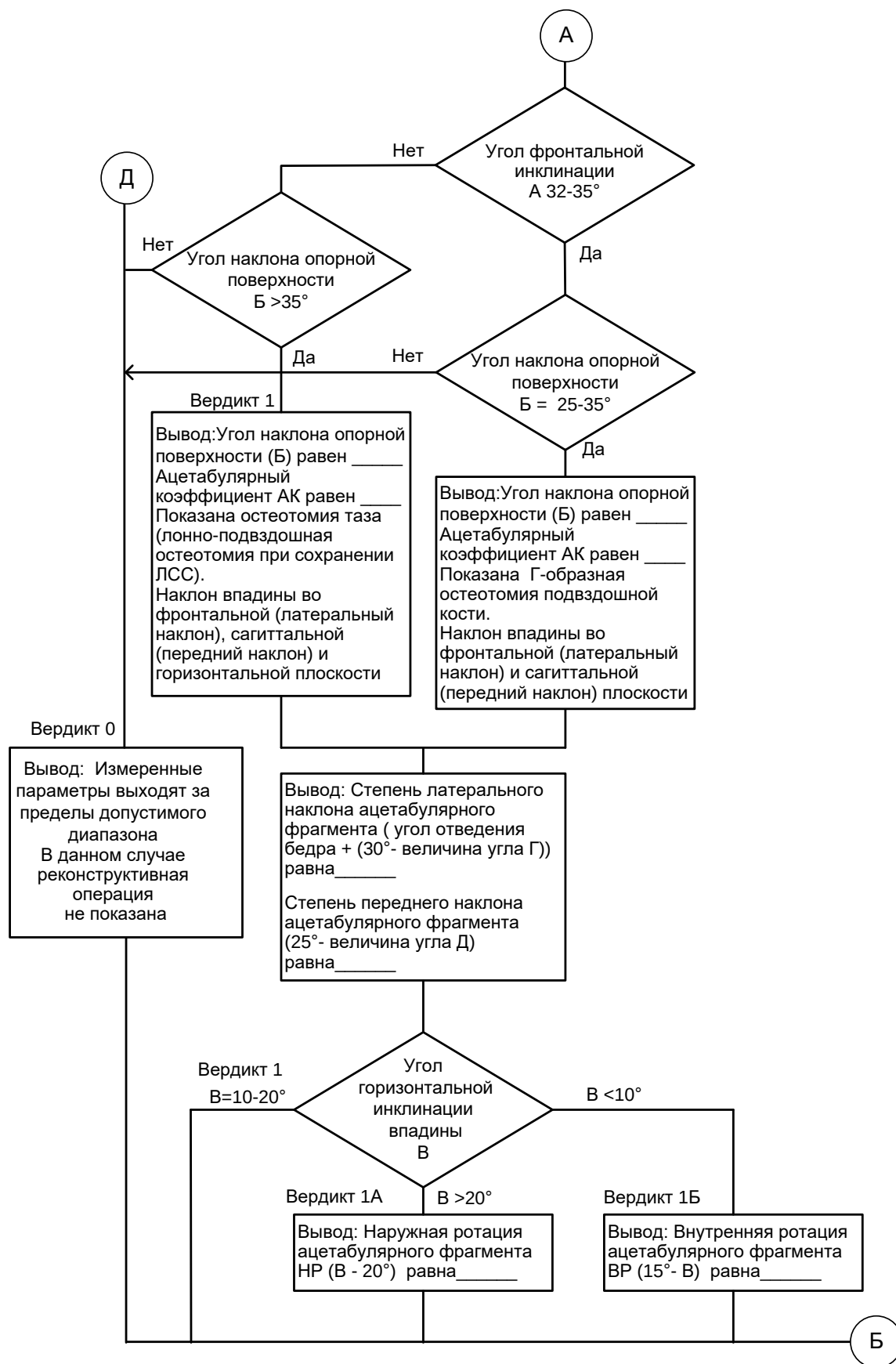


Рисунок 4 (б) – Возрастная группа от 7 до 9 лет.

Схема алгоритма (продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

24

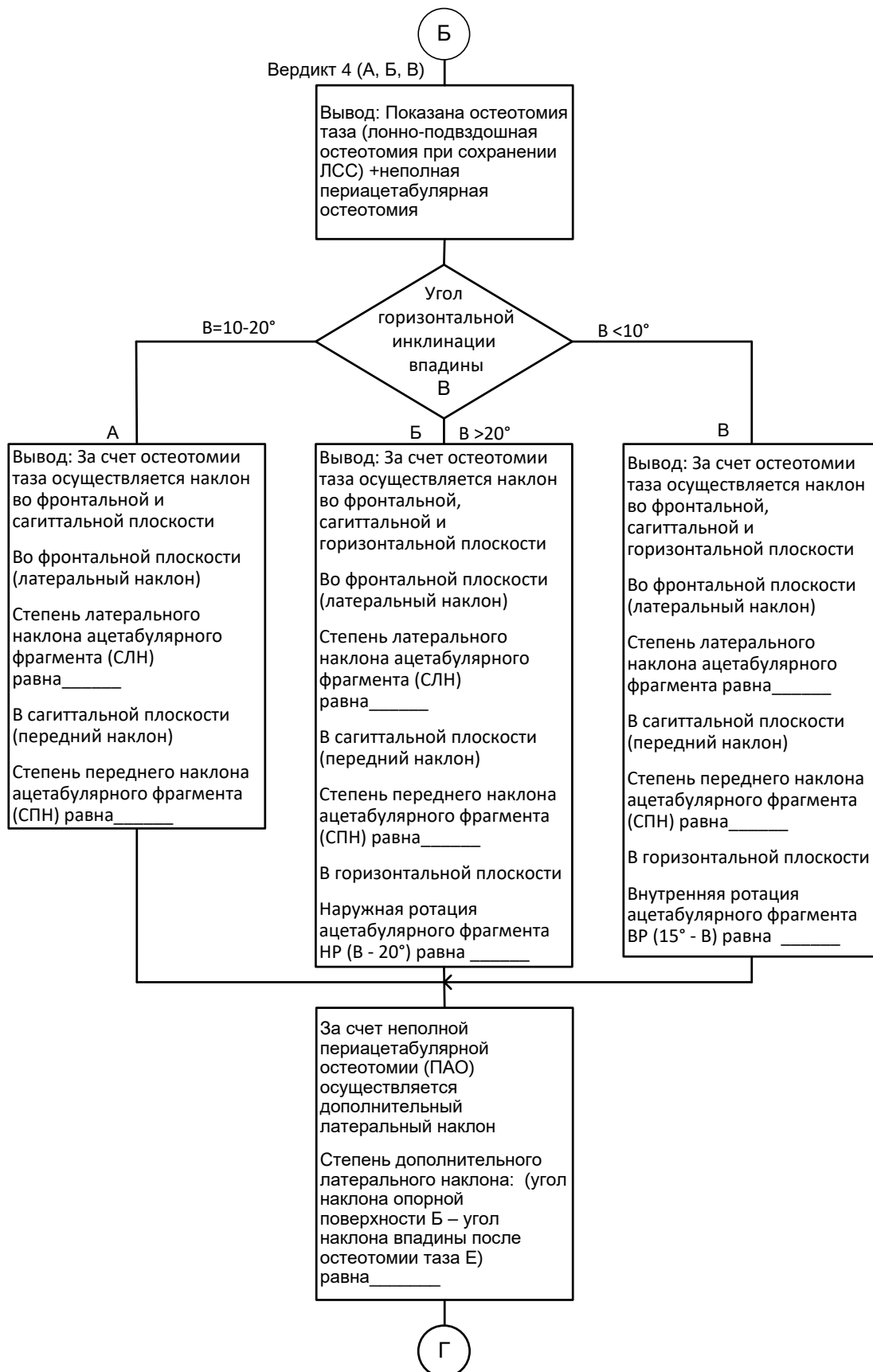


Рисунок 4 (в) – Возрастная группа от 7 до 9 лет.

Схема алгоритма (продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

25

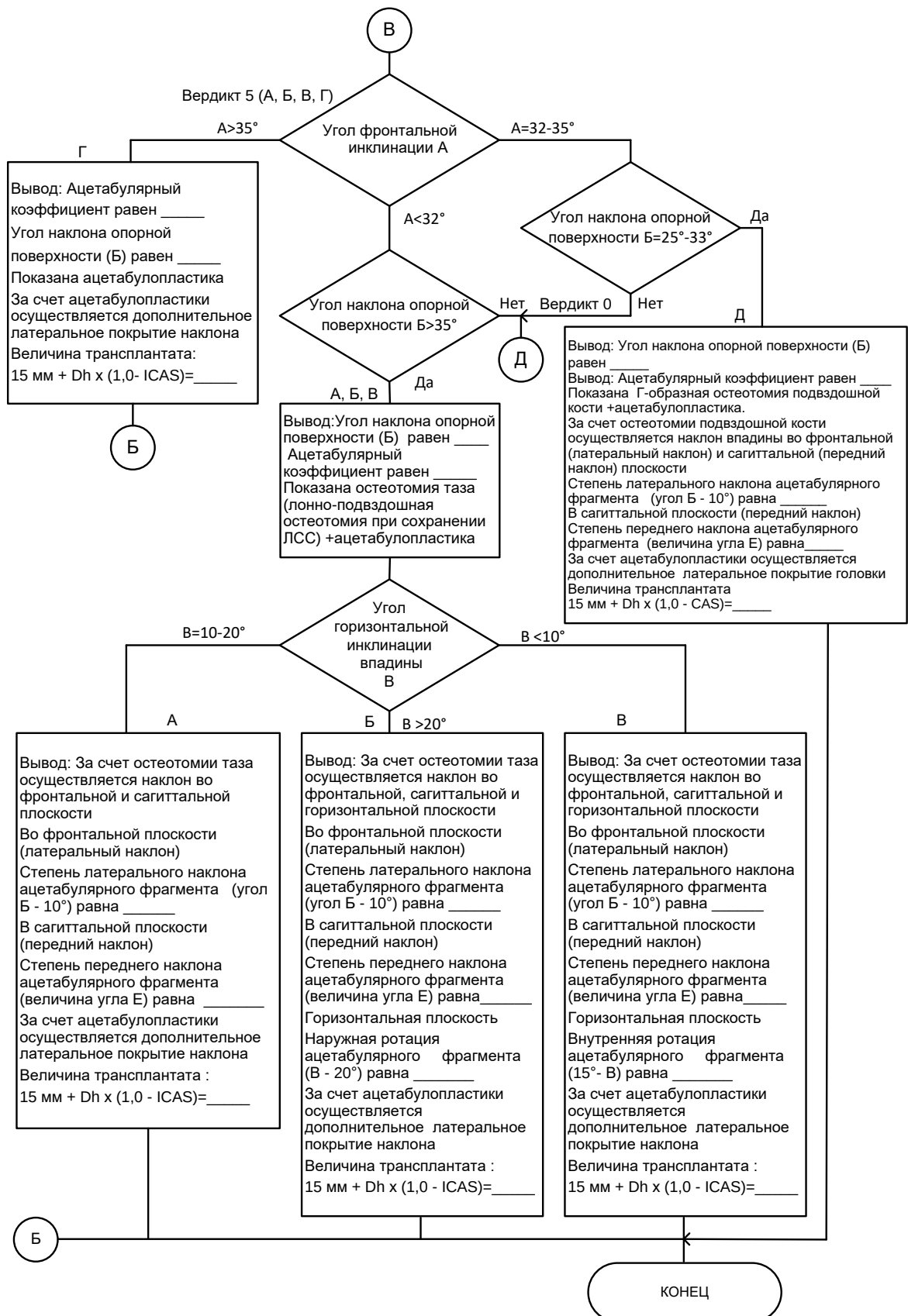


Рисунок 4 (г) – Возрастная группа от 7 до 9 лет.

Схема алгоритма (продолжение)

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

26

1.3 Разработка экранных форм для ввода параметров рентгенограмм

В соответствии с ТЗ, в ходе дипломного проектирования были разработаны девять экранных форм, предназначенных для ввода исходных параметров рентгенограмм пациента, по которым осуществляется определение расчетных параметров анализируемого участка опорно-двигательного аппарата и вынесение соответствующего вердикта, содержащего диагноз заболевания пациента и рекомендации по его дальнейшему лечению, а также для управления процессами расчета параметров рентгенограмм.

Все экранные формы имеют схожую структуру, выполнены в едином стиле, имеют примерно одинаковый состав органов управления и отображения информации.

Функции управляющих элементов просты и интуитивно понятны пользователю. Разработанные экранные формы приведены ниже, на рисунках 5-14 расчетно-пояснительной записки и будут рассмотрены далее, при описании основных функций разработанной программы.

1.3.1 Краткое описание экранных форм

Рассмотрим экранную форму «Окно 1», приведенную на рисунке 5. Из рисунка видно наличие на данной форме следующих элементов контроля, управления и отображения информации:

- кнопки управления «Выберете файл», «Помощь», «Измерьте диаметр головки Dh», «Измерьте расстояние между фигурами слезы R», «Рассчитать», «Далее», «Рентгенограмма 1», «Шпаргалка 1»;
- информационные панели «Загрузите рентгенограмму таза в прямой проекции», «Определение индекса сферичности головки ISh»;
- панели для вывода результатов измерений «Dh=____мм», «R=____мм», «Индекс сферичности головки ISh=Dh/(0,5R, ISh=____мм)».

Экранная форма «Окно 1» содержит максимальное количество элементов управления. На других формах, как это видно на рисунках, некоторые элементы управления не активны.

Коротко перечислим основные функции экранных форм, разработанных в ходе анализа ТЗ на проектирования программы:

- запуск. Пользователь запускает программу, первая экранная форма;
- загрузка. При нажатии кнопки «Выбрать файл» загружается требуемая рентгенограмма;
- измерения. Пользователь сначала нажимает кнопку «Измерить», рисует линии, углы или окружности на изображении, затем нажимает кнопку «Рассчитать» для получения параметров;
- переход. После получения требуемых данных пользователь нажимает кнопку «Далее» для перехода к следующему окну;
- выход. При нажатии кнопки «Выход» можно завершить работу с программой.

В ходе проектирования внешний вид разработанных экранных форм незначительно изменился, так как добавились некоторые дополнительные функции. Более подробное описание работы с экранными формами изложено ниже, в разделах «Руководство пользователя» и «Тестирование программы».

Окно №1 - Загрузка рентгенограммы №1 и определение индекса сферичности головки ISh

Рентгенограмма 1

Шпарталка 1

Выход

Загрузите рентгенограмму таза в прямой проекции

Выберете файл

Определение индекса сферичности головки ISh

Помощь

Измерьте диаметр головки Dh

Dh= _____ мм

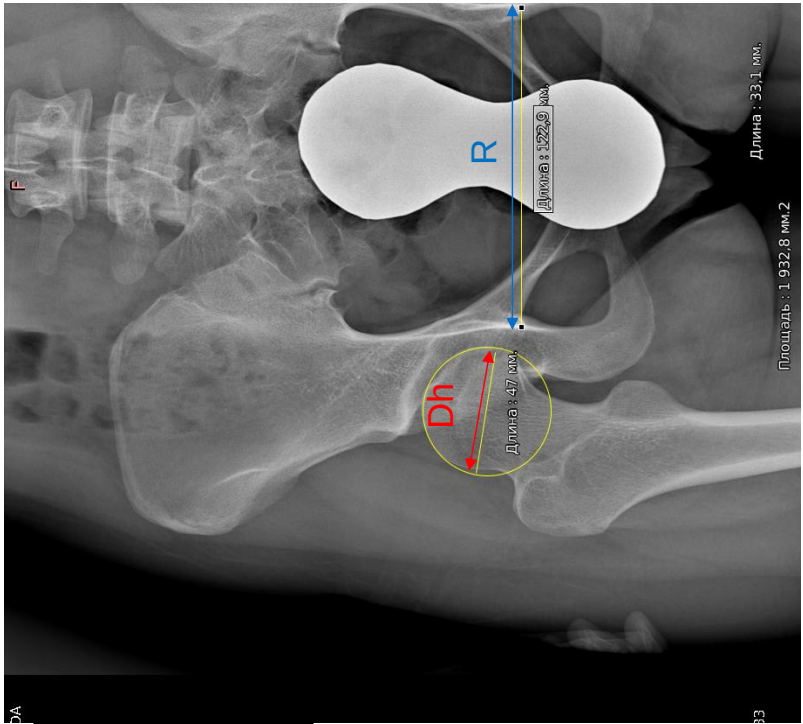
Измерьте расстояние между слезы R

R= _____ мм

Расчитать

Индекс сферичности головки ISh=Dh/(0.5R), ISh= _____

Далее



Индекс сферичности головки (ISh) - отношение диаметра окружности, соответствующей форме головки (Dh) к половине расстояния между фигурами слезы (R)

Рисунок 5 – Экранная форма ввода параметров и расчета индекса сферичности головки бедра (ISh)

Окно №2 - Определение индекса Окано

Рентгенограмма 1

Шпателька 2

Выход

Помощь

Измерьте расстояние A

Измерьте расстояние B

Индекс Окано: $(A/B) \cdot 100\% =$

A= _____ мм

B= _____ мм

Рассчитать

Далее

лев

Индекс сферичности головки по Окано (RI) – отношение расстояния от внутреннего края головки до линии, соответствующей наиболее проксимальной части головки (A) к расстоянию от внутреннего края до наружного края головки (B) (43-58%)

Рисунок 6 – Экранная форма ввода параметров и расчета индекса сферичности головки по Окано (RI)

Окно №3а - Определение углов фронтальной инклинации и наклона опорной поверхности

Рентгенограмма 1

Шпарталка 3

Шпарталка 4

Выход

Помощь

А=_____град

Б=_____град

Определение углов фронтальной инклинации и наклона опорной поверхности

Измерьте угол фронтальной инклинации А

Измерьте угол наклона опорной поверхности Б

Далее

Угол фронтальной инклинации **A** – угол между касательной, проведенной из фигуры слезы к наружному краю впадины, и вертикалью, проведенной из фигуры слезы

Рисунок 7 – Экранная форма ввода и расчета параметров угла фронтальной инклинации

Окно №36 - Определение углов фронтальной инклинации и наклона опорной поверхности

Рентгенограмма 1

Шпаргалка 3

Шпаргалка 4

Выход

Помощь

Измерьте угол фронтальной инклинации А

А=_____град

Измерьте угол наклона опорной поверхности В

В=_____град

Угол наклона опорной поверхности В - угол между горизонтальной линией, проведенной через нижний край зоны склероза (зона нагрузки) и касательной к краю впадины (0-10 °)

Далее

Рисунок 8 – Экранная форма ввода и расчета параметров угла наклона опорной поверхности

				Выход
	Рентгенограмма 1	Шаргалка 5		

Определение индекса сферичности впадины ISA

Измерьте диаметр впадины Da

Расстояние между фигурами слезы R

Рассчитать

Индекс сферичности впадины $ISA = Da / (0,5R)$, $ISA = \underline{\hspace{2cm}}$

Индекс конгруэнтности $ICAS = ISA / Ish$, $ICAS = \underline{\hspace{2cm}}$

Далее

Помощь

Da= _____ мм

R= _____ мм

Страница 1

Индекс сферичности впадины (ISA) - отношение диаметра окружности, соответствующей форме впадины к половине расстояния между фигурами слезы (0,8-1,24)

Рисунок 9 – Экранная форма ввода и расчета параметров
индекса сферичности впадины (ISA) и конгруэнтности (ICAS)

Окно №5 - Определение угла горизонтальной инклинации впадины

Рентгенограмма 1

Шаргалка 6

Выход

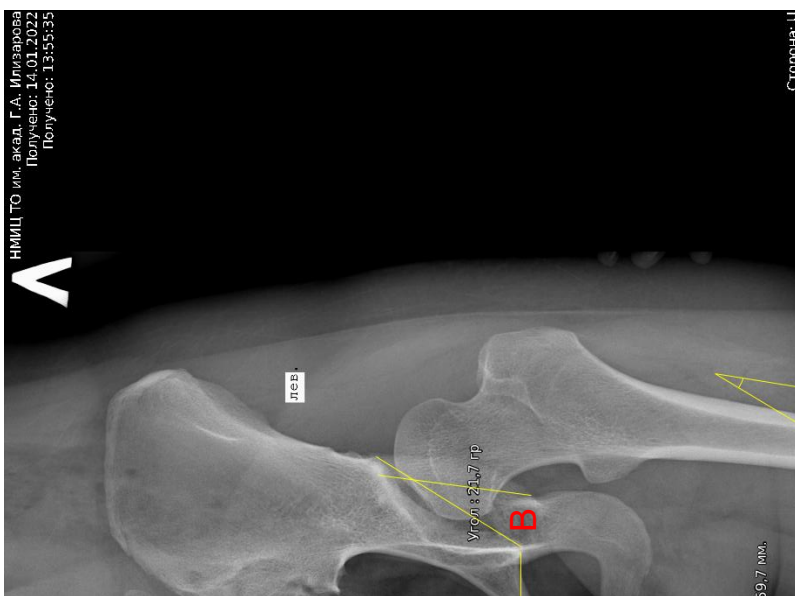
Определение угла горизонтальной инклинации впадины

Измерьте угол горизонтальной инклинации впадины В

Помощь

В= _____ град

Далее



Угол горизонтальной инклинации впадины В – угол между линией, соединяющей задне верхний край впадины с фигурой слезы и передне-верхний край впадины с задне нижним краем впадины (15-20°)

Рисунок 10 – Экранная форма ввода и расчета параметров угла горизонтальной инклинации впадины

Окно №6 - Определение ацетабулярного коэффициента

Рентгенограмма 1

Шаргалка 7

Выход

Определение ацетабулярного коэффициента АК

Помощь

Измерьте расстояние D

D= _____ мм

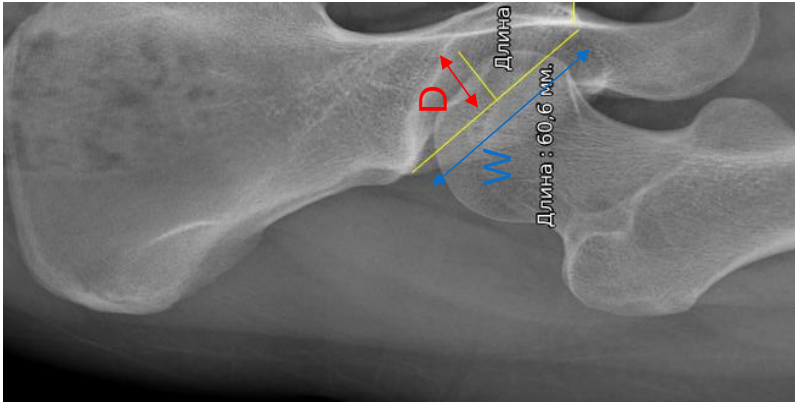
Измерьте расстояние W

W= _____ мм

Расчитать

Ацетабулярный коэффициент: (DW)* 1000= _____

Далее



Длина : 60,6 мм.

Длина

D

W

Ацетабулярный коэффициент (АК, отношение длины перпендикуляра (D), проведенного из центра впадины к линии, касательной ко входу во впадину, к длине отрезка, соединяющего фигуру «слезы» с верхненаружным краем впадины (W). АК=D/W x 1000; АК≥250)

Рисунок 11 – Экранная форма ввода и расчета параметров ацетабулярного коэффициента (АК)

Окно №7 - Загрузка рентгенограммы №2 и определение степени латерального наклона

Рентгенограмма 2

Шпателька 8

Выход

Загрузите рентгенограмму таза в положении отведения и внутренней ротации

Выберите файл

Определение степени латерального наклона ацетабулярного фрагмента

Помощь

Измерьте угол Виберга Г (норма 30°)

Г=_____град

Измерьте угол отведения бедра УОБ

УОБ=_____град

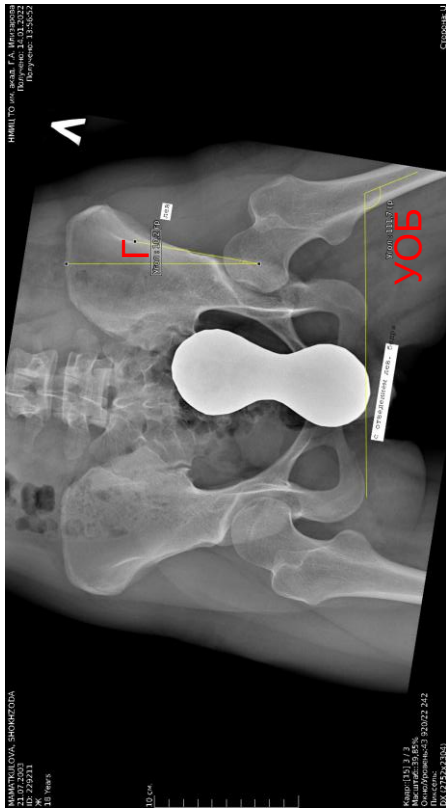
Расчитать

Угол Виберга (Г) – угол между линией проведенной через центр головки касательно к наружному краю впадины и вертикалью, проведенной через центр головки

Степень латерального наклона (СПН): УОБ+(30° - Г)=_____град

Далее

Угол отведения бедра (УОБ) - угол между горизонтальной линией, проведенной через нижний край зоны склероза (зона нагрузки) и осевой линией бедра в положении отведения и внутренней ротации



Угол Виберга (Г) – угол между линией проведенной через центр головки касательно к наружному краю впадины и вертикалью, проведенной через центр головки

Угол отведения бедра (УОБ) - угол между горизонтальной линией, проведенной через нижний край зоны склероза (зона нагрузки) и осевой линией бедра в положении отведения и внутренней ротации

Определение степени латерального наклона - определяется угол Виберга Г (норма 30°). Степень латерального наклона ацетабулярного фрагмента - угол отведения + (30° - величина угла Г)

Рисунок 12 – Экранная форма ввода и определения степени латерального наклона

Окно №8 - Загрузка рентгенограммы №3 и определение степени переднего наклона

Рентгенограмма 3

Шпартгалка 9

Выход

Загрузите рентгенограмму таза в ложно
профильной проекции

Определение степени переднего наклона

Измерьте передне краевой угол (Д, норма
25°)

Выберите файл

Помощь

Д=_____град

Рассчитать

Степень переднего наклона: 25° - Д=_____град

Далее



Угол Д – угол между линией проведенной через центр головки касательно к переднему краю впадины и вертикалью, проведенной через центр головки

Рисунок 13 – Экранная форма ввода и определения степени переднего наклона

Рентгенограмма 4

Шпателька 10

Выход

Загрузите рентгенограмму таза в аксиальной проекции

Выберите файл

Определение степени дополнительного латерального наклона

Помощь

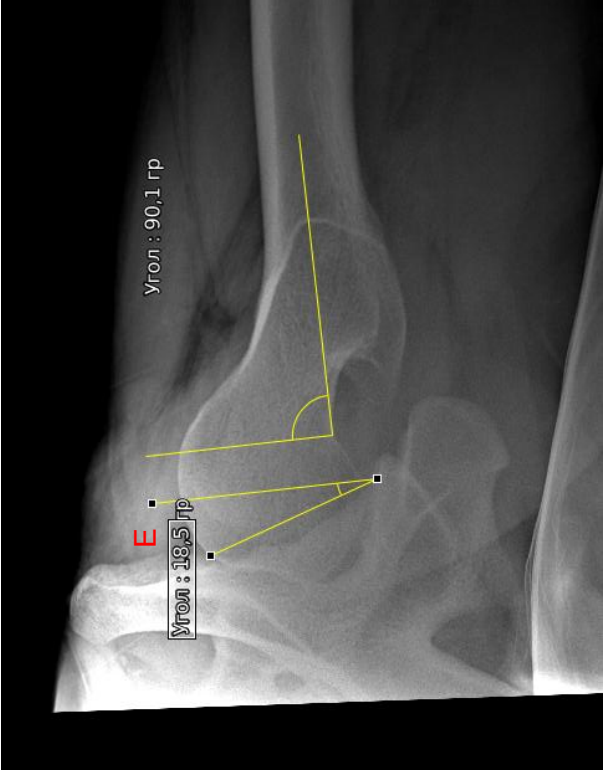
Измерьте угол E

E = _____ град

Степень дополнительного латерального наклона:
B - E = _____ град

Рассчитать

Далее



Угол : 18,5 гр

Угол : 90,1 гр

E

Определение степени дополнительного латерального наклона

Определяется угол (E) между касательной к краям впадины и перпендикуляром к оси проксимального отдела диафиза бедра (угол наклона впадины после остеотомии таза E)

Степень дополнительного латерального наклона: (угол наклона опорной поверхности B – угол наклона впадины после остеотомии таза E)

Рисунок 14 – Экранная форма ввода и определения степени дополнительного латерального наклона

2 Реализация программы автоматизации обработки рентгеновских снимков на языке высокого уровня

2.1 Выбор языка программирования и среды разработки

Выбор языка программирования C# и платформы .NET обусловлен рядом преимуществ [2, 3]. C# - это мощный объектно-ориентированный язык, в который входят множество библиотек, включая библиотеки для работы с графикой. Это значительно упрощает создание приложений с графическими интерфейсами.

Интеграция со средой разработки Visual Studio позволяет быстро и легко как разрабатывать новые программные продукты, так и тестировать [4 - 6].

Windows Forms [7] была выбрана как проверенная и достаточно быстрая платформа для создания настольных приложений, работающих на Windows. Она проста в освоении, позволяет быстро создавать прототипы интерфейсов и хорошо интегрируется с C#, что позволяет разработчику сосредоточиться на логике приложения.

Однако важно отметить некоторые ограничения выбранных технологий. C# является управляемым языком, поэтому он может быть не таким быстрым, как нативные языки, такие как C++, когда речь идет о задачах, требующих большого объема обработки изображений. Однако для задач интерактивного измерения на единичных снимках, не требующих сложных алгоритмов анализа в реальном времени, его быстродействия оказывается более чем достаточно, а выигрыш в скорости разработки и надежности кода перевешивает этот аспект. Windows Forms - это более старая платформа, которая не предлагает столько возможностей для настройки внешнего вида вашего приложения или для ускорения графики с помощью аппаратного обеспечения. Новые технологии, такие как WPF или MAUI, лучше справляются с этими задачами [8]. Однако для создания функционального и удобного интерфейса медицинского приложения,

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		39

где приоритетом являются удобство и точность выполнения конкретных задач, а не изысканный дизайн, его возможностей вполне достаточно.

Таким образом, сочетание C# и Windows Forms будет хорошим и сбалансированным выбором для решения поставленной задачи.

2.2 Описание программы автоматизации обработки рентгеновских снимков

Программа «Medic» — это Windows Forms приложение, разработанное на C# для анализа рентгенограмм таза. Оно предназначено для загрузки изображений рентгенограмм, выполнения измерений различных параметров (углов, расстояний, диаметров) и генерации вердикта о необходимости хирургического вмешательства на основе этих данных. Программа имеет модульную структуру, состоящую из нескольких окон (форм), каждое из которых отвечает за определённый этап анализа, а также вспомогательных файлов для управления данными и интерфейсом.

2.2.1 Общая структура программы

Программа состоит из следующих основных файлов:

- Program.cs — точка входа в приложение;
- SharedData.cs — статический класс для хранения общих данных;
- Window_(1-9).cs — файлы с логикой для каждого окна программы;
- Window_(1-9).Designer.cs — файлы с описанием отображения элементов интерфейса.

Каждое окно (Window_1–Window_9) представляет собой отдельный этап анализа рентгенограмм, где пользователь загружает изображения, выполняет измерения и переходит к следующему этапу. Данные между окнами передаются через класс SharedData.

2.2.2 Описание файлов и их назначение

2.2.2.1 Файл Program.cs

Назначение.

Файл Program.cs является точкой входа в приложение, где запускается программа. Этот файл необходим для инициализации приложения и управления его жизненным циклом.

Функции и логика.

Метод Main настраивает визуальные стили приложения, регистрирует обработчик события ApplicationExit, который вызывает методы освобождения ресурсов изображений из SharedData, а также запускает первое окно программы (Window_1).

2.2.2.2 Файл SharedData.cs

Назначение.

Файл SharedData.cs является статическим классом для хранения данных, общих для всех окон программы. Обеспечивает централизованное хранение данных, доступных из всех окон, и управляет ресурсами изображений.

Свойства.

Radiograph_1, Radiograph_2, Radiograph_3, Radiograph_4 — объекты Bitmap для хранения загруженных рентгенограмм.

Переменные для сохранения измеренных параметров:

- Dh (диаметр головки);
- Da (диаметр ацетабулярного фрагмента);
- R (расстояние между фигурами слезы);
- ISh (индекс сферичности головки бедра);
- Dist_A, Dist_B, Dist_D, Dist_W (различные измеряемые расстояния);
- Okano (индекс Окано);
- Angle_A, Angle_B, Angle_V, Angle_E (измеряемые углы);
- ISA (индекс сферичности ацетабулярного фрагмента)*
- ICAS (индекс конгруэнтности, соответствия);
- AK (ацетабулярный коэффициент);
- UOB, SLN, SPN_1, SDLN, HP, BP, trans (дополнительные параметры);
- PHIO (ФИО);

- year (год рождения);
- age (возраст пациента).

2.2.2.3 Файлы Window_(1-9).Designer.cs

Назначение.

Файлы данного типа содержат код, описывающий элементы интерфейса. Код определяет внешний вид и расположение элементов интерфейса, упрощая разработку посредством дизайнера форм.

Компоненты.

Метод InitializeComponent() — инициализирует и настраивает:

- элементы интерфейса (кнопки, метки, текстовые поля, PictureBox и т.д.);
- переменные для элементов интерфейса (например, btn_select_1, pictureBox1, textBox_1_1_Dh);
- настройки свойств элементов (размеры, шрифты, расположение, обработчики событий).

2.2.2.4 Файлы Window_(1-9).cs

Назначение.

Данные файлы содержат логику работы каждого окна и его элементов, включая обработку событий, измерения и расчеты.

Общая структура окон:

- каждое окно наследуется от Form и реализует определённый этап анализа;
- включает методы для загрузки изображений, переключения между рентгенограммой и шпаргалкой, отображения помощи, выполнения измерений и перехода к следующему окну.

2.2.2.5 Повторяющиеся элементы интерфейса и логика их работы

Элементы интерфейса.

Кнопка "Выбрать файл" (btn_select_*):

- открывает диалоговое окно для загрузки изображения рентгенограммы;
- используется в четырёх окнах.

Все подобные кнопки работают следующим образом:

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		42

```

private void btn_select_1_Click(object sender, EventArgs e)
// код, который будет выполняться при нажатии кнопки «Выбрать
файл» в окне №1
{
    using (OpenFileDialog openFileDialog = new OpenFileDialog())
// открыть меню выбора файла
    {
        openFileDialog.Title = "Выберите изображение";
// заголовок меню
        openFileDialog.Filter = "Файлы
изображений|*.png;*.jpg;*.jpeg|PNG файлы|*.png|JPEG
файлы|*.jpg;*.jpeg";
// какие файлы можно выбирать

        if (openFileDialog.ShowDialog() == DialogResult.OK)
// когда пользователь выберет файл
        {
            try
            {
                SharedData.Radiograph_1 =
(Bitmap) Image.FromFile(openFileDialog.FileName);
// картинка сохраниться в контейнерном классе SharedData
                pictureBox1.Image = SharedData.Radiograph_1;
// картинка отобразится на экране

                btn_switch_1.Enabled = true;
// включить кнопку «Рентгенограмма 1»
                btn_switch_2.Enabled = (clueImage_1 != null);
// включить кнопку «Шпаргалка 1»
            }
            catch (Exception ex)
// если произойдёт ошибка то появится сообщение об этом
            {
                MessageBox.Show($"Ошибка при загрузке
изображения: {ex.Message}",

```

```

        "Ошибка",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
}
}.

```

Кнопки переключения изображений (btn_switch_*_1, btn_switch_*_2) выполняют следующие функции:

- переключают между рентгенограммой и шпаргалкой;
- присутствуют во всех окнах (например, btn_switch_1_1, btn_switch_1_2 в

Window_1).

Все подобные кнопки работают следующим образом:

```

private void btn_switch_1_Click(object sender, EventArgs e)
// код, который будет выполняться при нажатии кнопки
«Рентгенограмма 1» в окне №1
{
    if (SharedData.Radiograph_1 != null)
// если рентгенограмма была загружена
    {
        pictureBox1.Image = SharedData.Radiograph_1;
// на экране отобразится рентгенограмма 1
    }
}
private void btn_switch_2_Click(object sender, EventArgs e)
// код, который будет выполняться при нажатии кнопки «Шпаргалка
1» в окне №1
{
    pictureBox1.Image = clueImage_1;
// на экране отобразится шпаргалка 1
    drawnPoints.Clear();
// очистить массив точек, который используется для рисования
    pictureBox1.Invalidate();
}

```

```
// перерисовать окно, чтобы стереть нарисованное ранее  
}.
```

Кнопка "Помощь" (btn_help_*):

- показывает/скрывает дополнительную информацию;
- присутствует во всех окнах (например, btn_help_1 в Window_1).

Все подобные кнопки работают следующим образом:

```
private void btn_help_1_Click(object sender, EventArgs e)  
// код, который будет выполняться при нажатии кнопки «Помощь» в  
окне №1  
{  
    label_1_3.Visible = !label_1_3.Visible;  
// переключить видимость текстовой информации (по умолчанию  
невидима)  
}.
```

Кнопка "Измерить" (btn_measure_*):

- активирует режим рисования для измерений;
- пример: btn_measure_1_1, btn_measure_1_2 в Window_1.

Все подобные кнопки работают следующим образом:

```
private void btn_measure_1_1_Click(object sender, EventArgs e)  
// код, который будет выполняться при нажатии кнопки «Измерить»  
в окне №1  
{  
    if (SharedData.Radiograph_1 != null)  
// если файл рентгенограммы №1 загружен  
    {  
        if (pictureBox1.Image != SharedData.Radiograph_1)  
// если на экране не отображается рентгенограмма 1  
        {  
            MessageBox.Show("Рентгенограммы 1 нет на экране!",  
// выводится сообщение об ошибке  
                "Ошибка",  
                MessageBoxButtons.OK,  
                MessageBoxIcon.Error);  
        }  
    }  
}
```

```

    }
    else
// если на экране отображается рентгенограмма 1
    {

// Сброс состояния и включение режима рисования
        isDrawingMode = true;
        isFirstPointSet = false;
        textBox_1_2_R.Text = "0";

// Очистка предыдущего рисунка
        if (drawingLayer != null)
        {
            drawingLayer.Dispose();
            drawingLayer = null;
        }

// Перерисовка PictureBox
        pictureBox1.Invalidate();
    }
    else
// если файл рентгенограммы №1 не загружен
    {
        MessageBox.Show("Сначала загрузите рентгенограмму!",
// выводится сообщение об ошибке
            "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}

```

Кнопка "Рассчитать" (btn_measure_*):

- вычисляет параметры на основе измерений;

- пример: btn_measure_1_3 в Window_1.

Все подобные кнопки работают следующим образом:

```
private void btn_measure_1_3_Click(object sender, EventArgs e)
// код, который будет выполняться при нажатии кнопки
«Рассчитать» в окне №1
{
    if (textBox_1_1_Dh.Text == "" || textBox_1_2_R.Text == "")
// если требуемые величины не измерены
    {
        MessageBox.Show("Требуемые значения не измерены!", //
            "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
    else
// если требуемые величины измерены
    {
        SharedData.Dh = Convert.ToDouble(textBox_1_1_Dh.Text);
// величина Dh сохраняется как число
        SharedData.R = Convert.ToDouble(textBox_1_2_R.Text);
// величина R сохраняется как число
        SharedData.ISh = (SharedData.Dh / (SharedData.R * 0.5))
// расчёт ISh
        textBox_1_3_ISh.Text =
Convert.ToString((float)SharedData.ISh); // отображение ISh на
экране в текстовом поле
    }
}.
```

Кнопка "Далее" (btn_next_*):

- осуществляет переход к следующему окну после проверки заполненности данных;

- пример: btn_next_1 в Window_1.

Все подобные кнопки работают следующим образом:

```
private void btn_next_3_Click(object sender, EventArgs e)
// код, который будет выполняться при нажатии кнопки «Далее» в
окне №3
{
    if (textBox_3_1_Angle_A.Text == "" ||
    textBox_3_2_Angle_B.Text == "")

// если требуемые значения не измерены
    {
        MessageBox.Show("Требуемые значения не измерены!",
// выводится сообщение об ошибке
            "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
    else
// если требуемые значения измерены
    {
        SharedData.Angle_A =
Convert.ToDouble(textBox_3_1_Angle_A.Text);
// измеренные величины сохраняются как число
        SharedData.Angle_B =
Convert.ToDouble(textBox_3_2_Angle_B.Text);
// измеренные величины сохраняются как число
        Window_4 w4 = new Window_4();
// создаётся окно №4
        w4.Show();
// отображается окно №4
        Hide();
// скрывается окно №3
    }
}
```

Окно отображения рентгенограмм (PictureBox (pictureBox*):

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		48

- отображает рентгенограмму и позволяет рисовать линии/окружности;
- пример: pictureBox1 в Window_1.

Текстовые поля (textBox_*):

- выводят результаты измерений и расчетов;
- пример: textBox_1_1_Dh, textBox_1_3_ISh в Window_1.

Рассмотрим применение основных функций создания и обработки графических изображений.

2.2.2.6 Рисование линий

Функция рисования линии позволяет пользователю нарисовать линию на изображении, указав две точки. Процесс начинается с активации режима рисования нажатием кнопки «Измерить». После этого пользователь кликает по изображению в PictureBox, чтобы задать начальную и конечную точки линии.

Рисование линии осуществляется при выполнении следующих процедур:

- активация режима рисования;
- первый клик;
- движение мыши;
- второй клик;
- отрисовка.

Рассмотрим подробно реализацию данных процедур.

Активация режима рисования.

Нажатие кнопки включает режим рисования, сбрасывает предыдущие данные (например, точки и слой рисования) и готовит PictureBox к взаимодействию.

При нажатии кнопки btn_measure выполняется следующая процедура.

```
isDrawingMode = true;
isFirstPointSet = false;
// Сброс предыдущих данных
drawingLayer?.Dispose();
drawingLayer = null;
pictureBox1.Invalidate();
```

При этом;

- сбрасывается состояние точек (isFirstPointSet = false);
- удаляется предыдущий слой рисования;
- активируется перерисовка PictureBox.

Первый клик.

Когда пользователь кликает мышью, координаты клика преобразуются из экранных в координаты изображения, и эта точка сохраняется как начальная. В этот момент программа начинает следить за движением мыши.

Обработка в PictureBox1_MouseDown.

```
Point imagePoint = GetImageCoordinates(e.Location);
firstPoint = imagePoint;
// Фиксация стартовой точки
isFirstPointSet = true;
currentMousePosition = imagePoint;
// Координаты преобразуются в систему изображения
через GetImageCoordinates(); Точка сохраняется в firstPoint.
```

Движение мыши.

Пока пользователь перемещает курсор, от начальной точки до текущего положения курсора отображается временная красная линия. Это происходит за счёт перерисовки PictureBox при каждом движении мыши.

Обработка в PictureBox1_MouseMove.

```
currentMousePosition = GetImageCoordinates(e.Location);
double length = CalculateDistance(firstPoint,
currentMousePosition);
textBox_1_2_R.Text = Math.Round(length, 2).ToString();
// Обновление значения
pictureBox1.Invalidate();
// Принудительная перерисовка. Рассчитывается расстояние между
точками в реальном времени; Значение отображается в текстовом
поле
```

Второй клик.

При втором клике фиксируется конечная точка. Временная линия заменяется финальной, которая рисуется на специальном слое — объекте Bitmap, называемом drawingLayer. Этот слой накладывается на исходное изображение, чтобы не изменять его.

В PictureBox1_MouseDown.

```
CreateFinalLine(imagePoint);  
// Фиксация конечной точки  
isDrawingMode = false;  
// Отключение режима  
Метод CreateFinalLine():  
drawingLayer = new Bitmap(pictureBox1.Image.Width,  
pictureBox1.Image.Height);  
using (Graphics g = Graphics.FromImage(drawingLayer))  
{  
    g.DrawLine(pen, firstPoint, secondPoint);  
// Постоянная линия  
    g.FillEllipse(Brushes.Red, firstPoint.X-3,  
firstPoint.Y-3, 6, 6); // Точки  
    g.FillEllipse(Brushes.Red, secondPoint.X-3,  
secondPoint.Y-3, 6, 6);  
}
```

Отрисовка.

Финальная линия рисуется красным цветом с толщиной 2 пикселя, а на концах линии добавляются красные точки для наглядности. Длина линии вычисляется как расстояние между двумя точками и отображается в текстовом поле.

В PictureBox1_Paint.

```
if (isDrawingMode && isFirstPointSet)  
{  
    e.Graphics.DrawLine(pen, displayFirst,  
displayCurrent); // Временная линия  
// Отрисовка точек (аналогично финальной версии)  
}
```

Временная линия рисуется напрямую на Graphics элемента управления;

Финальная линия сохраняется в drawingLayer.

Особенности координат: Метод GetImageCoordinates() корректирует позицию с учётом масштабирования:

```
int imageX = (int)((pictureBoxPoint.X - offsetX) / scale);  
int imageY = (int)((pictureBoxPoint.Y - offsetY) / scale);
```

Особенности.

Для корректной работы с разными размерами изображения и PictureBox используются преобразования координат. Это позволяет рисовать линии точно там, где кликнул пользователь, независимо от масштаба изображения.

2.2.2.7 Рисование окружности

Функция рисования окружности активируется кнопкой «Измерить» и позволяет построить окружность по двум точкам, которые задают её диаметр. Процесс похож на рисование линии, но с другими вычислениями.

Рисование окружности осуществляется при выполнении следующих процедур.

Активация режима.

Нажатие кнопки «Измерить» включает режим рисования окружности, сбрасывает предыдущие данные и подготавливает PictureBox.

Отличается установкой флага окружности:

```
isDrawingMode = true;  
isDrawingCircleMode = true;  
// Специфичный флаг .
```

Первый клик.

Пользователь задаёт первую точку диаметра, которая фиксируется в координатах изображения.

Движение мыши.

При перемещении курсора отображается временная окружность. Её центр вычисляется как середина между первой точкой и текущим положением курсора,

а радиус — как половина расстояния между этими точками. Временная окружность и диаметр рисуются красным цветом.

В PictureBox1_MouseMove:

```
double diameter = CalculateDistance(firstPoint,
currentMousePosition);
textBox_1_1_Dh.Text = Math.Round(diameter, 2).ToString();
// Текущий диаметр.
```

Второй клик.

При втором клике фиксируется вторая точка диаметра. Финальная окружность рисуется на слое drawingLayer с использованием тех же вычислений: центр — середина отрезка, радиус — половина длины диаметра. Диаметр отображается в текстовом поле.

В PictureBox1_Paint для временной окружности:

```
DrawCircleByDiameter(e.Graphics, pen, displayFirst,
displayCurrent);
// Метод DrawCircleByDiameter():
int centerX = (p1.X + p2.X) / 2;
// Центр = середина диаметра
int centerY = (p1.Y + p2.Y) / 2;
float radius = (float)(diameter / 2);
g.DrawEllipse(pen, centerX-radius, centerY-radius, radius*2,
radius*2);
g.DrawLine(pen, p1, p2);
// Линия диаметра.
```

Отрисовка.

На слое рисуется окружность, линия диаметра, красные точки на концах диаметра и синяя точка в центре для наглядности. При втором клике вызывается CreateFinalCircle():

```
DrawCircleByDiameterOnImage(g, pen, firstPoint, secondPoint);.
```

Реализация аналогична DrawCircleByDiameter(), но рисует на слое drawingLayer.

Особенности.

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		53

Как и при рисовании линии, используется слой `drawingLayer`, чтобы сохранить оригинальное изображение неизменным. Преобразование координат обеспечивает точность при масштабировании.

2.2.2.8 Измерение угла

Функция измерения угла позволяет пользователю задать три точки на изображении, чтобы измерить угол между двумя линиями, сходящимися в вершине. Процесс запускается кнопкой «Измерить» и проходит в несколько этапов.

Измерение угла осуществляется при выполнении следующих процедур:

- активация режима;
- первый клик;
- второй клик;
- третий клик;
- динамическое отображение;
- отрисовка;
- вычисление угла.

Рассмотрим подробно реализацию данных процедур.

Активация режима.

Нажатие кнопки переводит приложение в режим ожидания первой точки, очищает предыдущие данные и готовит `PictureBox` к рисованию.

Механизм.

При нажатии кнопки «Измерить» (напр. `btn_measure_3_1`) вызывается метод `StartMeasurement()`, который:

- сбрасывает предыдущие измерения (`drawnPoints.Clear()`);
- устанавливает начальную фазу `WaitingForP1`;
- привязывает текстовое поле для вывода результата (`activeAngleTextBox`);
- инициализирует перо для рисования (`new Pen(Color.Yellow, 3)`).

```
private void StartMeasurement(TextBox targetTextBox)
{
    currentPhase = DrawingPhase.WaitingForP1;
```

```

        drawnPoints.Clear();
        activeAngleTextBox = targetTextBox;
        activeAngleTextBox.Text = "";
        pictureBox3.Invalidate();
    }

```

Первый клик.

Пользователь задаёт первую точку, которая станет началом первой линии. После этого приложение переходит в режим ожидания второй точки.

Механизм.

При клике в режиме WaitingForP1 координаты сохраняются в коллекцию drawnPoints, фаза переключается на WaitingForP2:

```

        case DrawingPhase.WaitingForP1:
            drawnPoints.Add(e.Location);
// Сохранение точки
            currentPhase = DrawingPhase.WaitingForP2;
            currentCursorPoint = e.Location;
// Инициализация для динамической линии
            break;.

```

Второй клик.

Фиксируется вторая точка, которая становится вершиной угла. От первой точки к текущему положению курсора рисуется первая линия, а при движении мыши — вторая временная линия от вершины.

Механизм.

Второй клик (фаза WaitingForP2) добавляет вершину угла и активирует динамическое рисование второй линии:

```

        case DrawingPhase.WaitingForP2:
            drawnPoints.Add(e.Location); // Фиксация вершины
            currentPhase = DrawingPhase.WaitingForP3;
            currentCursorPoint = e.Location;
            break;.

```

Третий клик.

Задаётся третья точка, завершающая вторую линию. После этого угол фиксируется, а процесс рисования завершается.

Механизм.

Третий клик (фаза WaitingForP3) фиксирует конечную точку линии и вычисляет угол:

```
case DrawingPhase.WaitingForP3:
    drawnPoints.Add(e.Location);
    currentPhase = DrawingPhase.DrawingCompleted;
    UpdateAngleDisplay(true); // Финализация расчёта
    break;
```

Динамическое отображение.

Во время движения мыши после второго клика отображается вторая линия и дуга, показывающая угол между первой линией и текущим положением курсора. Угол обновляется в текстовом поле в реальном времени.

Визуализация.

При движении мыши после фиксации вершины:

- рисуется временная линия от вершины до курсора;
- рассчитывается и отображается текущий угол;
- дуга угла обновляется в реальном времени.

```
private void pictureBox3_MouseMove(object sender,
MouseEventArgs e)
{
    currentCursorPoint = e.Location;
    if (currentPhase == DrawingPhase.WaitingForP3)
    {
        UpdateAngleDisplay(false); // Динамический расчёт
        pictureBox3.Invalidate(); // Принудительная
        перерисовка
    }
}
```

Отрисовка.

Линии рисуются жёлтым цветом с толщиной 3 пикселя. Дуга угла строится с фиксированным радиусом и отображает измеряемый угол. После третьего клика все элементы фиксируются на экране.

Графика.

Используется антиалиасинг для сглаживания.

В методе Paint:

- линии рисуются жёлтым цветом (Pen(Color.Yellow, 3));
- дуга угла строится с радиусом ARC_RADIUS = 30f.

```
e.Graphics.SmoothingMode = SmoothingMode.AntiAlias; //
Качественный рендеринг
e.Graphics.DrawLine(drawingPen, drawnPoints[0],
drawnPoints[1]); // Линия 1
e.Graphics.DrawLine(drawingPen, drawnPoints[1],
currentCursorPoint); // Линия 2
DrawAngleArc(e.Graphics, ...); // Отрисовка дуги
```

Вычисление угла.

Угол рассчитывается с использованием скалярного произведения векторов, образованных тремя точками. Результат переводится в градусы и отображается с точностью до трёх знаков после запятой.

Используется алгоритм скалярного произведения векторов.

Ключевые операции:

- вычисление векторов от вершины угла;
- расчёт косинуса через скалярное произведение;
- преобразование в градусы с точностью 0.001.°

```
public static double CalculateAngleBetweenPoints(PointF
p0, PointF pVertex, PointF pEndArm)
{
    double v1x = p0.X - pVertex.X;
    double v1y = p0.Y - pVertex.Y;
    double v2x = pEndArm.X - pVertex.X;
    double v2y = pEndArm.Y - pVertex.Y;
```

```

        double dotProduct = v1x * v2x + v1y * v2y;
        double mag1 = Math.Sqrt(v1x * v1x + v1y * v1y);
        double mag2 = Math.Sqrt(v2x * v2x + v2y * v2y);
        double cosTheta = dotProduct / (mag1 * mag2);
        double angleRad = Math.Acos(Math.Max(-1.0,
Math.Min(1.0, cosTheta)));
        return angleRad * (180.0 / Math.PI);
    }
    // Конвертация в градусы
    // Вывод в TextBox: activeAngleTextBox.Text =
angle.ToString("F3");

```

Особенности: процесс управляется через этапы (фазы), что позволяет точно отслеживать, сколько точек уже задано. Динамическое обновление линий и дуги обеспечивает визуальную обратную связь для пользователя.

3 Руководство пользователя

Программа «Medic» предназначена для анализа рентгенограмм и имеет понятный интерфейс.

3.1 Краткое описание функций программы

Кратко перечислим основные функции разработанной в ходе дипломного проектирования программы:

- запуск. Пользователь запускает программу, открывается Window_1;
- загрузка. При нажатии кнопки «Выбрать файл» загружается требуемая рентгенограмма;
- измерения. Пользователь сначала нажимает кнопку «Измерить», рисует линии, углы или окружности на изображении, затем нажимает кнопку «Рассчитать» для получения параметров;
- переход. После получения требуемых данных пользователь нажимает кнопку «Далее» для перехода к следующему окну;
- финал. В последнем (девятом) окне вводятся данные пациента, измеряется угол E, и генерируется вердикт, сохраняемый в RTF-файле.

3.2 Подробное описание функций программы

3.2.1 Открытие рентгенограммы

Нажатием кнопки «Выбрать файл» открывается стандартное меню выбора файла в Windows, где требуется указать путь к файлу рентгенограммы. Рентгенограмма должна быть в формате PNG или JPG (JPEG). До выбора файла кнопки переключения между рентгенограммой и визуальной подсказкой недоступны.

3.2.2 Получение текстовых подсказок

При нажатии кнопки «Помощь» в левом нижнем углу появляется текстовое поле, которое содержит описание требуемых действий или объяснение расчёта коэффициентов.

3.2.3 Измерение длин

Для начала измерения длины требуется нажать кнопку «Измерить» у тех значений, которые требуют наличие длин. После нажатия кнопки для измерения длины следует в окне рентгенограммы нажать левой кнопкой мыши в точке начала измерения, после установки первой точки будет визуально видна линия или диаметр вместе с окружностью следующая за курсором, начало которого будет в первой точке. Далее следует переместить курсор в конечную точку измерения, после чего нужно еще раз нажать левую кнопку мыши. Значение автоматически заполнится в текстовое поле под кнопкой. Линия измерения или диаметр с окружностью останутся на экране рентгенограммы до тех пор, пока не будет осуществлено новое измерение или не будет произведён переход к «Шпаргалке». Если потребуется произвести измерение заново, то это можно сделать повторным нажатием кнопки «Измерить».

3.2.4 Измерение углов

Процесс почти аналогичен измерению длин, за исключением того, что требуется сделать три нажатия левой кнопкой мыши. После начала измерения следует выбрать первую точку нажатием левой кнопкой мыши в поле с рентгенограммой. Должна появиться линия, следующая за курсором мыши. Аналогичным образом выбирается вторая точка. После выбора второй точки кроме того, что появляется фиксированная линия между первой и второй точкой, а также линия идущая от второй линии к курсору, между двумя формируемыми линиями появляется дуга. Измерение завершается третьим кликом мыши в требуемой точке, затем значение угла автоматически заполняется в текстовое поле ниже. Если потребуется произвести измерение заново, то это можно сделать повторным нажатием кнопки «Измерить».

3.2.5 Переход к «Шпаргалке»

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		60

Переход к визуальной инструкции можно осуществить нажатием кнопки «Шпаргалка i », где i – номер шпаргалки, зависящий от типов коэффициентов, которые требуется измерить в данный момент. После перехода к «шпаргалке» в поле рентгенограммы появятся текстовые и визуальные указания с примером измерений, а также стираются все линии, которые были оставлены от измерений. Возврат к рентгенограмме осуществляется при помощи кнопки «Рентгенограмма k », где k – номер рентгенограммы. Файлы содержащие визуальные подсказки хранятся в папке вместе с программой и поставляются вместе с ней и могут быть просмотрены при помощи встроенного программного обеспечения Windows или другими графическими редакторами.

3.2.6 Кнопка «Рассчитать»

Выполняет расчёт значений коэффициентов из измеренных значений со снимка. Заполняет их в текстовое поле ниже. Расчёт можно провести заново повторным нажатием кнопки.

3.2.7 Кнопка «Далее»

Позволяет перейти к следующему окну для измерения других требуемых коэффициентов. Перед нажатием данной кнопки следует убедиться в правильности измерений, так как возврат назад не является возможным. Если никаких измерений не произведено или значения коэффициентов выходят за пределы требуемого диапазона возникает окно с ошибкой.

3.2.8 Ввод данных пациента

Перед формированием вердикта требуется ввести данные в поля «ФИО:» - полное имя пациента, «Годрождения» - год рождения, «Кол-во полных лет» - количество полных лет. Начало ввода осуществляется нажатием левой кнопкой мыши по полю, а затем осуществляется ввод данных с клавиатуры.

3.2.9 Формирование вердикта

После заполнения данных пациента вердикт формируется кнопкой «Заверить». Если требуемые поля заполнены, то после нажатия возникает окно с

сообщением, что файл сформирован в папке с программой. Файл создается в формате RTF для дальнейшего просмотра или печати пользователем.

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		62

4 Тестирование программы

За неимением реальных рентгенограмм были использованы отредактированные рисунки с «шпаргалок», в которых отсутствует текст подсказок, а также указания наименований длин и углов, измеряемых величин и некоторые измерительные линии. В программу была загружена первая рентгенограмма (см. рисунок 15).

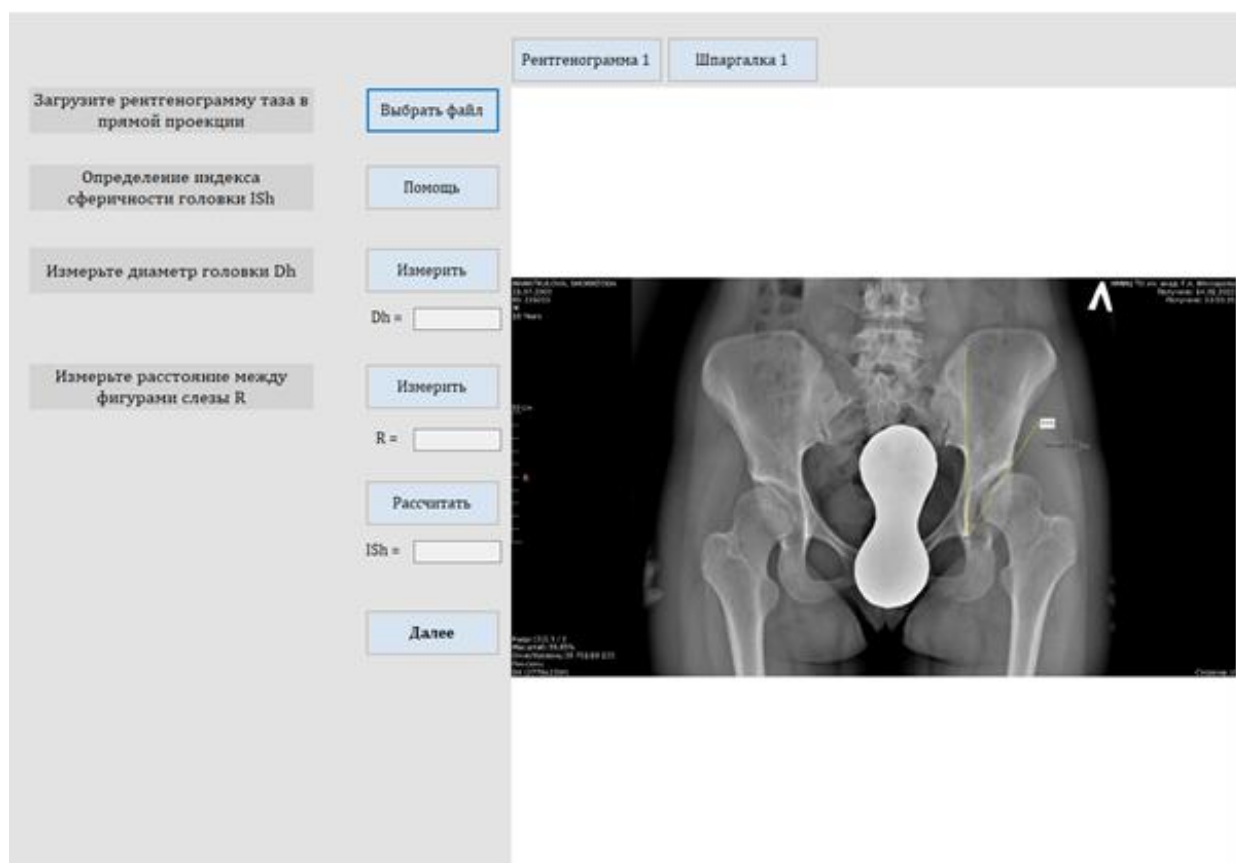


Рисунок 15 – Окно программы после открытия первой рентгенограммы

Был выполнен переход к «шпаргалке» при помощи нажатия кнопки «Шпаргалка 1» (см. рисунок 16).

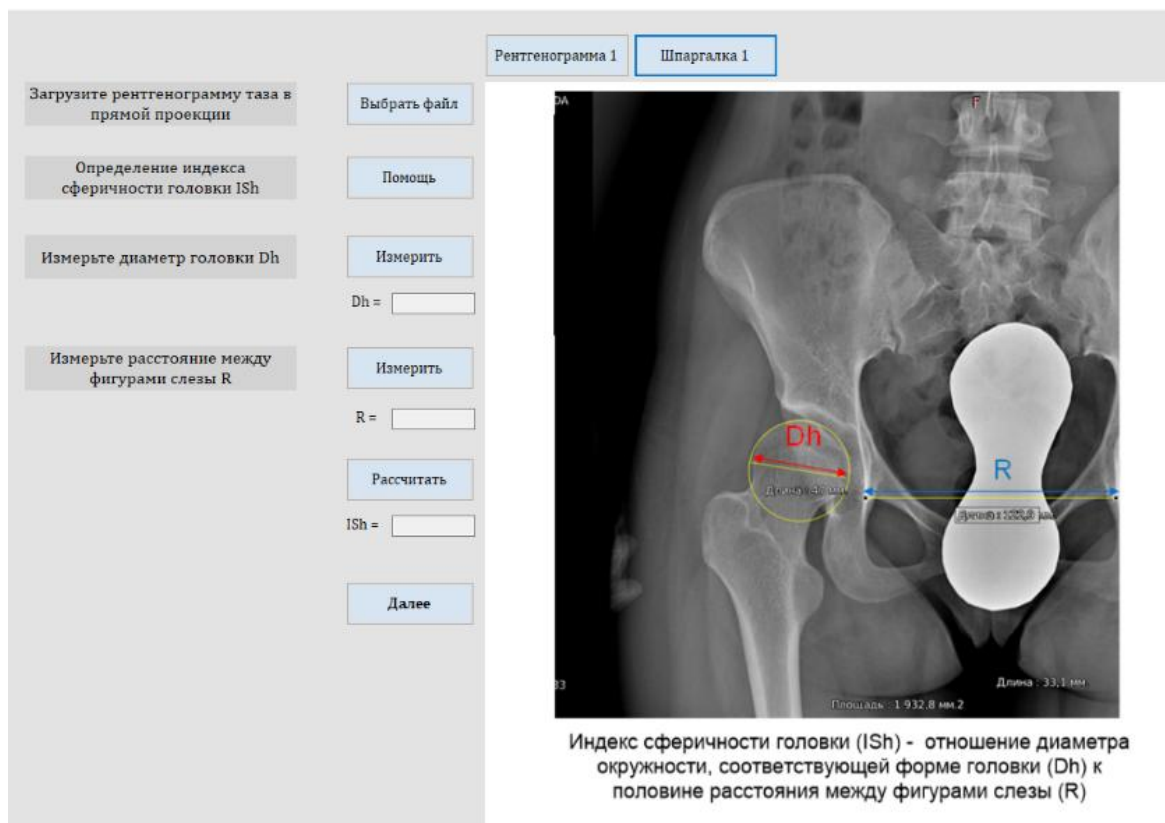


Рисунок 16 – Окно программы после открытия первой шпаргалки

После изучения информации был выполнен обратный переход при помощи кнопки «Рентгенограмма 1» и были выполнены соответствующие измерения и расчёт ISh (см. рисунок 17).

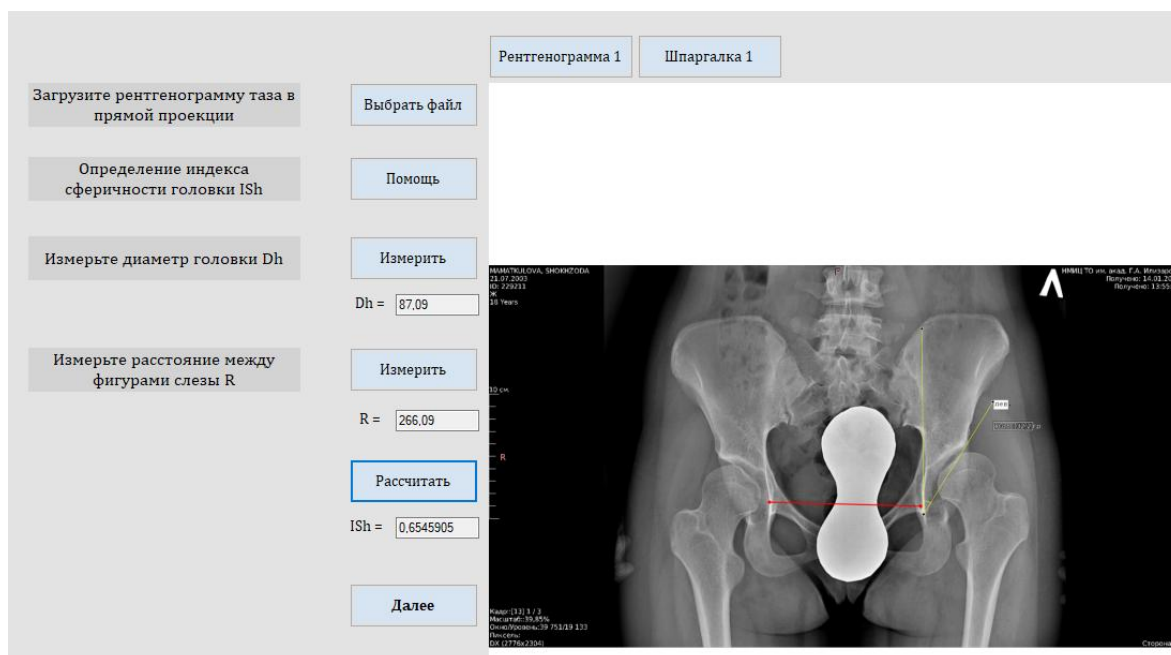


Рисунок 17 – Окно программы после измерения и расчёта ISh

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

64

После измерений был осуществлён переход на следующий экран, где была открыта очередная «шпаргалка» (см. рисунок 18).

Рентгенограмма 1
Шпаргалка 2

Определение индекса Окано

Измерьте расстояние А

Измерьте расстояние В

А =

В =

Рассчитать

Индекс Окано

Далее

лев

Индекс сферичности головки по Окано (RI) – отношение расстояния от внутреннего края головки до линии, соответствующей наиболее проксимальной части головки (А) к расстоянию от внутреннего края до наружного края головки (В) (43-58%)

Рисунок 18 – Окно программы после открытия второй шпаргалки

После изучения информации был осуществлён переход обратно к рентгенограмме и были выполнены соответствующие измерения и расчёт индекса Окано. Был произведен переход на следующий экран (см. рисунок 19).

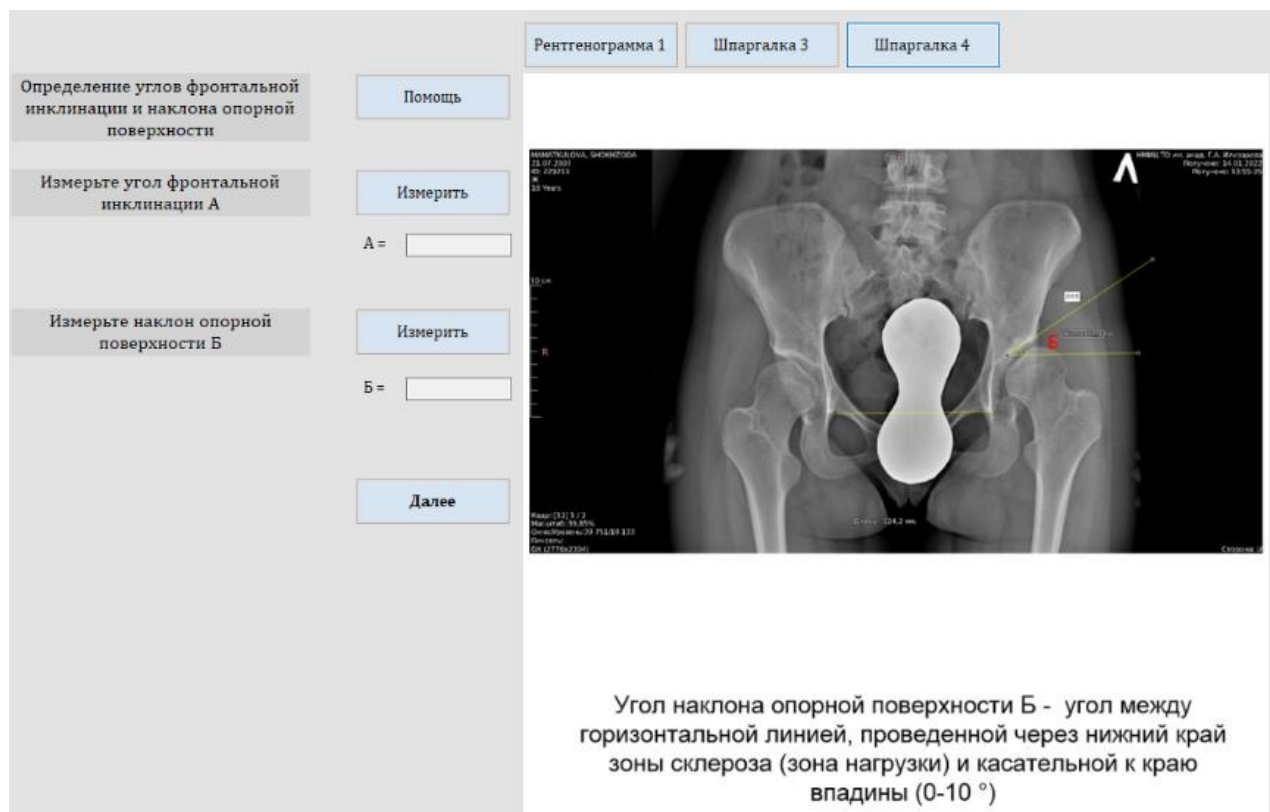


Рисунок 21 – Окно программы после открытия четвертой шпаргалки, измерение угла А

Были выполнены измерения углов А и Б. После чего был осуществлён переход к следующему экрану (см. рисунок 22).

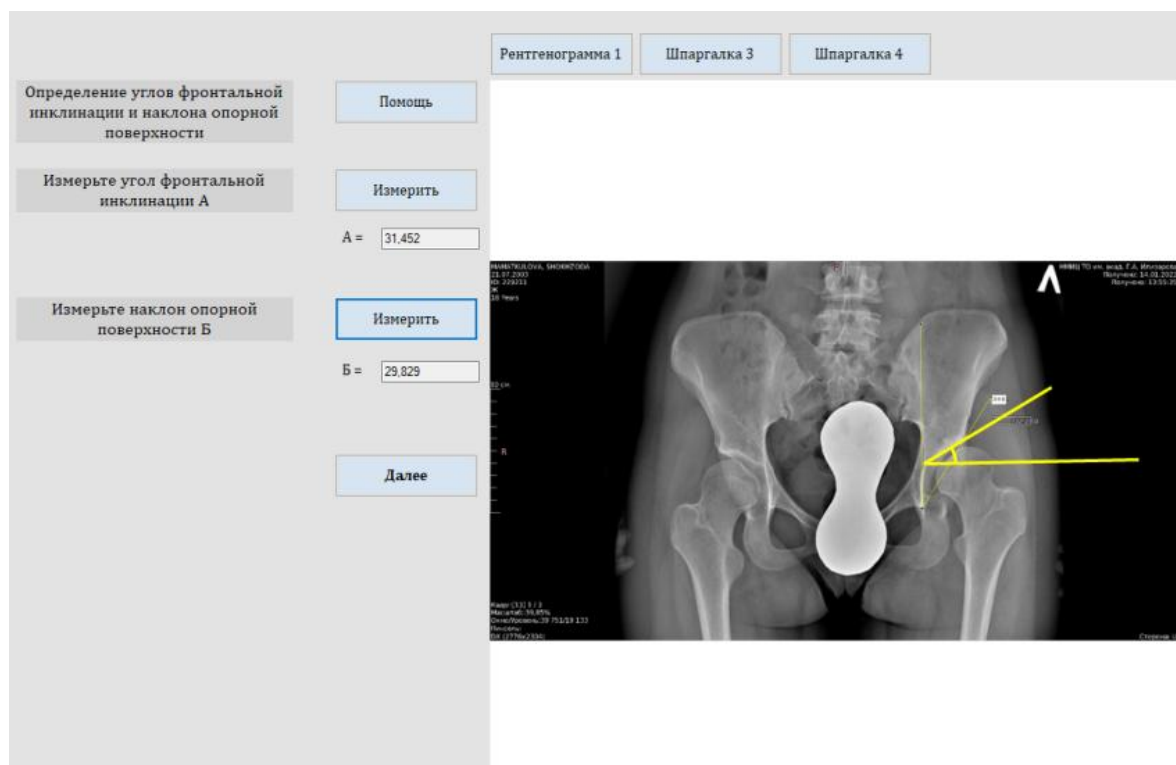


Рисунок 22 – Окно программы после измерения углов А и Б.

Была открыта «шпаргалка» 5 и изучена информация с неё. После чего был осуществлён возврат к рентгенограмме (см. рисунок 23).

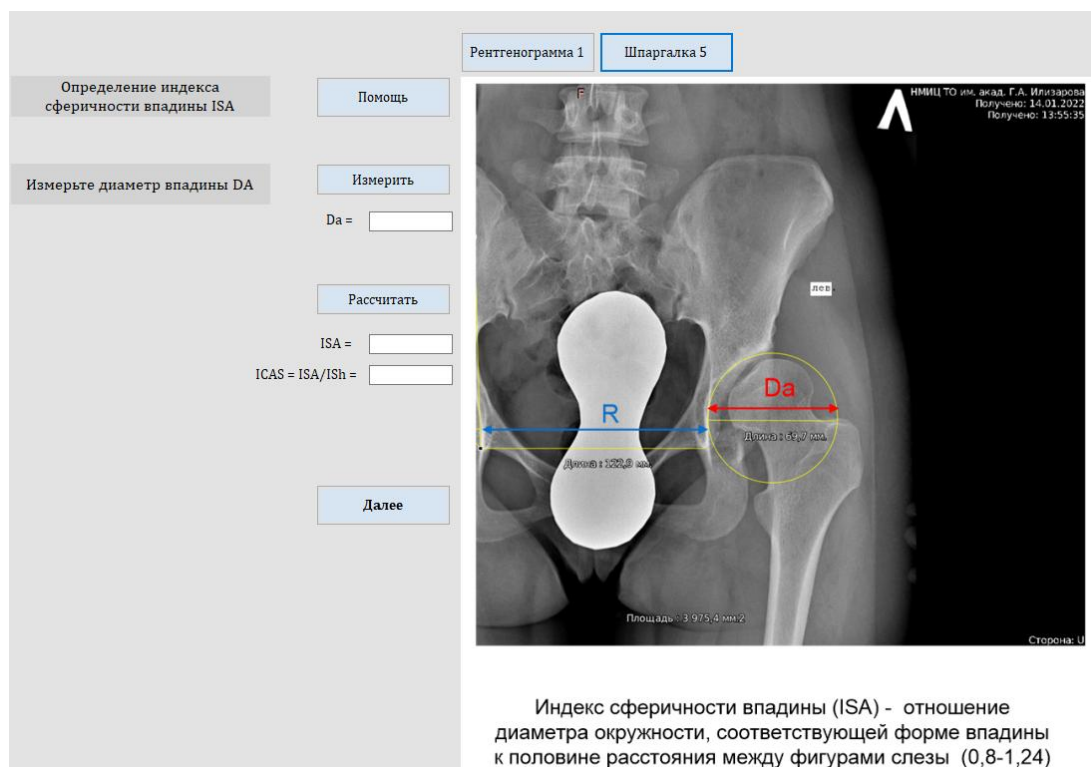


Рисунок 23 – Окно программы после открытия пятой шпаргалки

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

68

Были выполнены соответствующие измерения и выполнен расчёт ISA и ICAS. Затем был выполнен переход на следующий экран (см. рисунок 24).

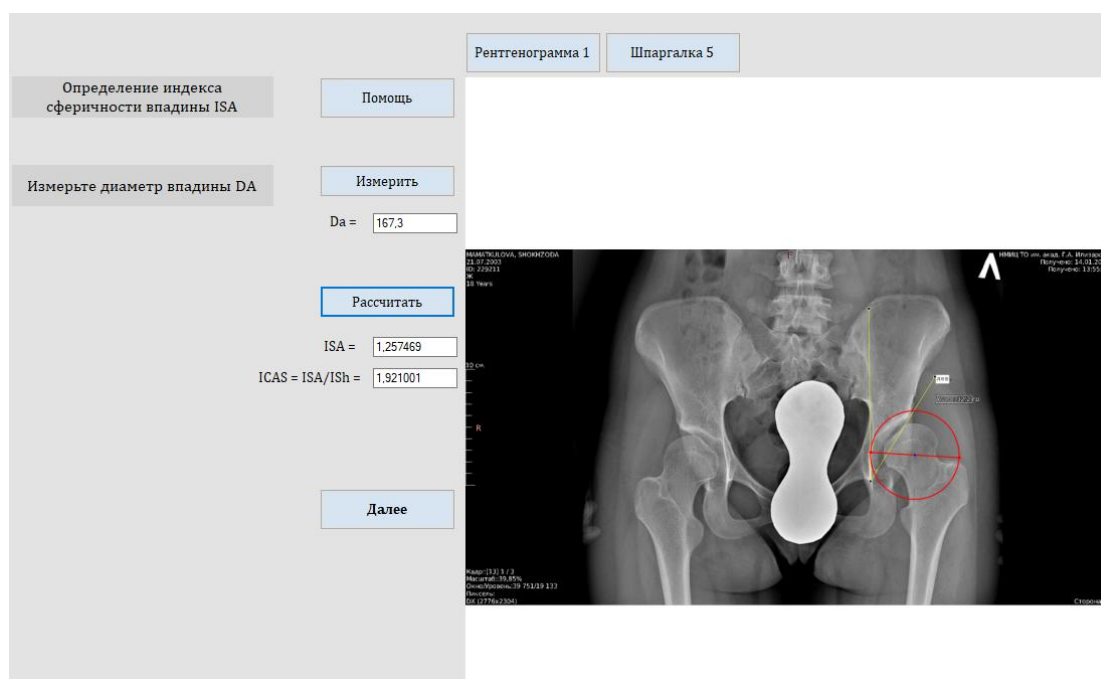


Рисунок 24 – Окно программы после измерения и расчёта ISA и ICAS

Была открыта «шпаргалка» 6 и изучена информация с неё (см. рисунок 25).

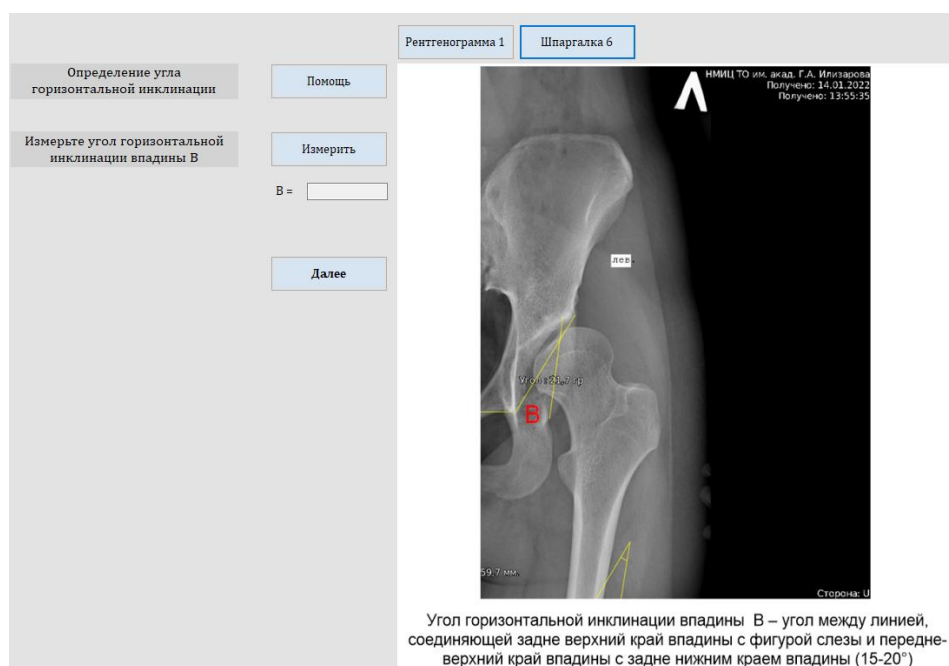


Рисунок 25 – Окно программы после открытия шестой шпаргалки

Далее было выполнено измерение угла В и осуществлён переход в следующее окно.

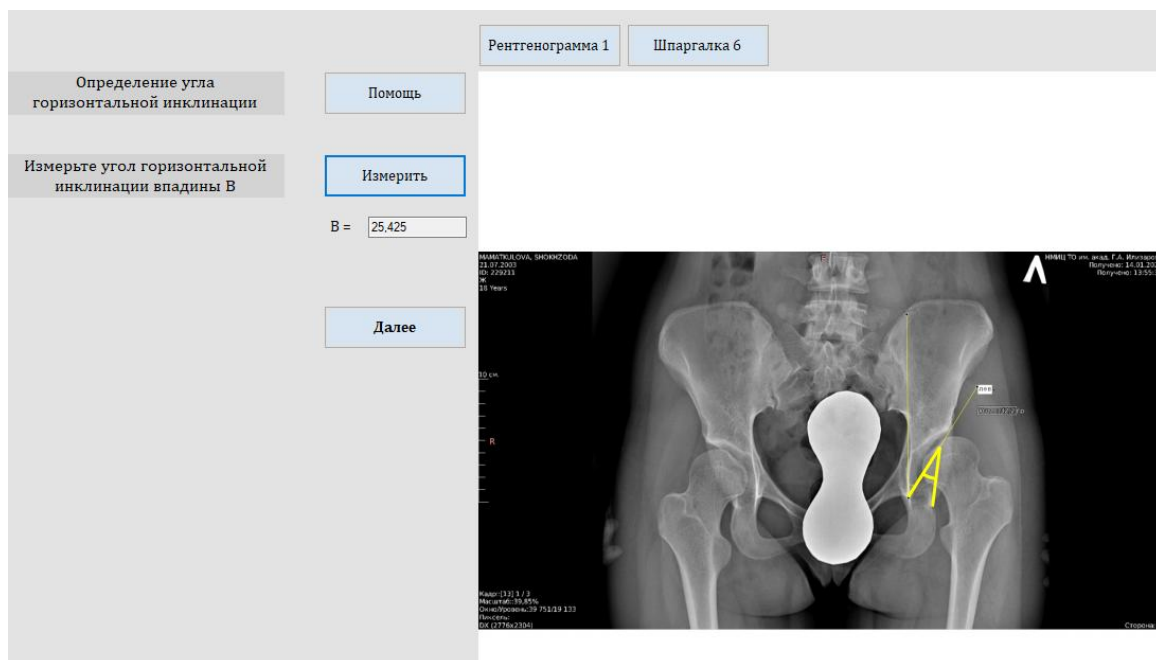


Рисунок 26 – Окно программы после измерения угла В

Далее была изучена информация с «шпаргалки» 7 (см. рисунок 27), после чего был осуществлён возврат в окно с рентгенограммой (см. рисунок 28), где были выполнены соответствующие измерения и расчёт АК.

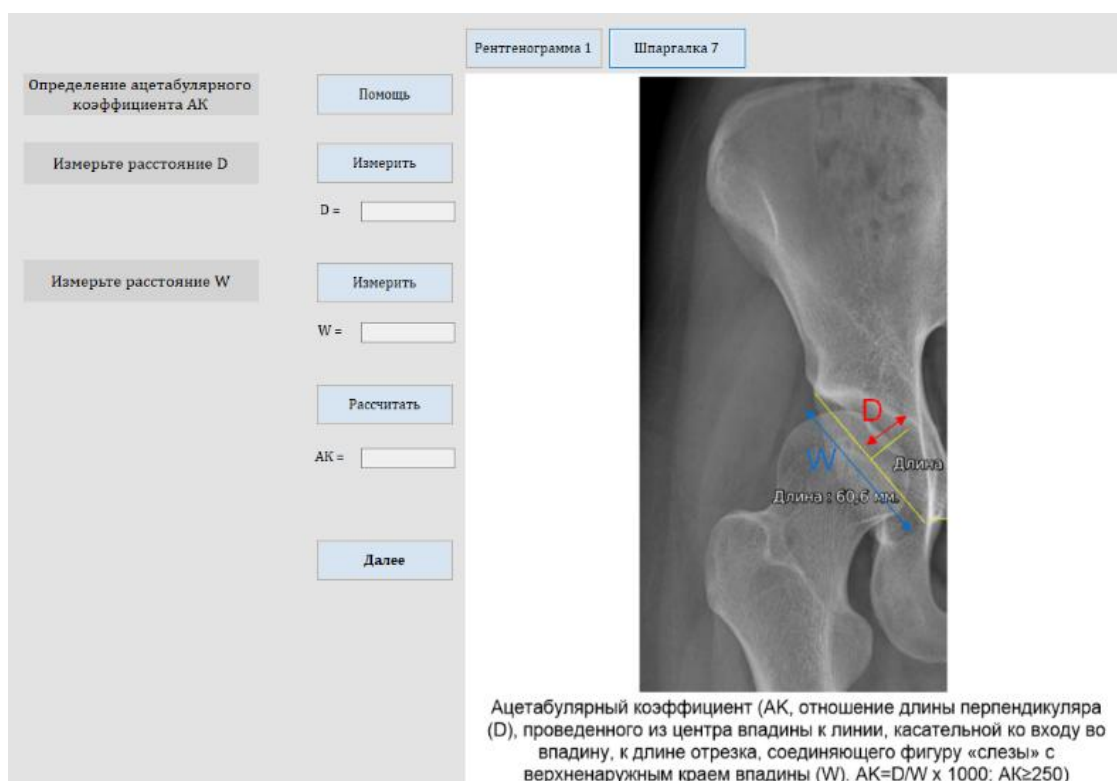


Рисунок 27 – Окно программы после открытия седьмой шпаргалки

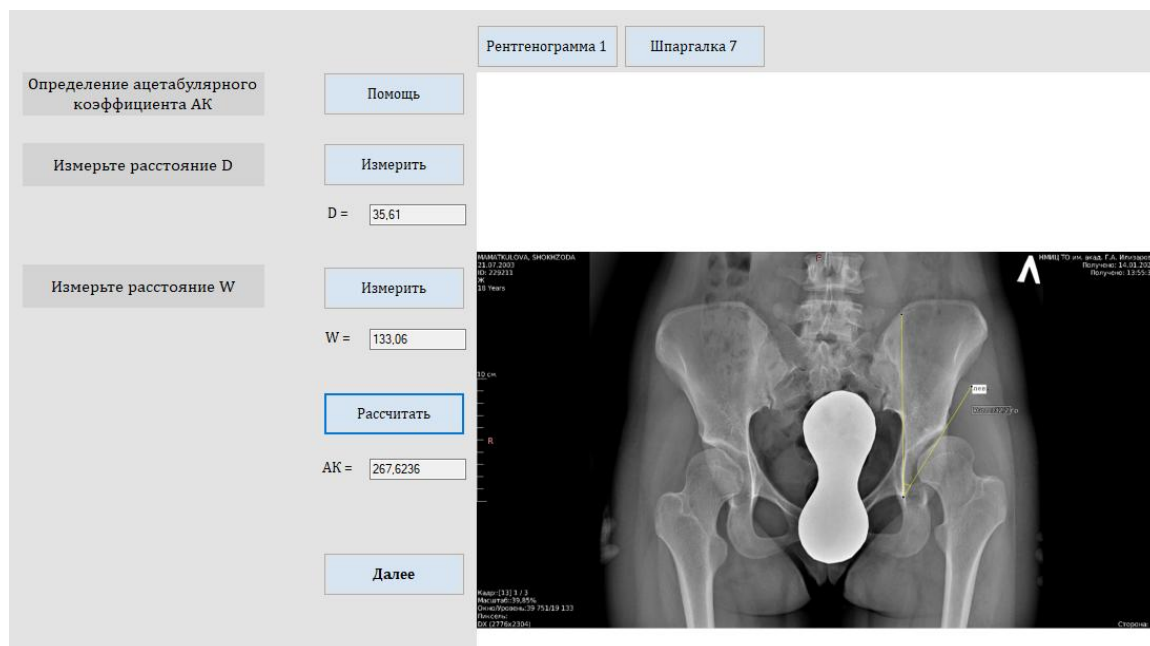


Рисунок 28 – Окно программы после измерения и расчёта АК.

Далее была загружена рентгенограмма 2 и открыта «шпаргалка» 8 для изучения информации (см. рисунок 29).

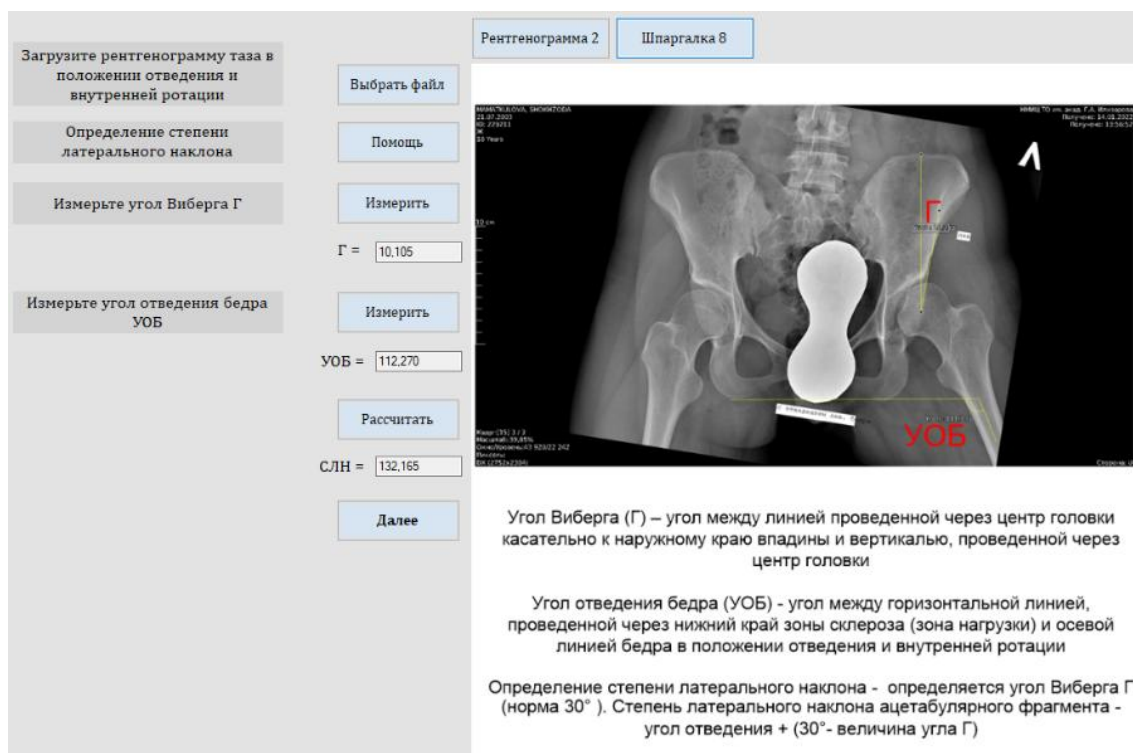


Рисунок 29 – Окно программы после открытия восьмой шпаргалки

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

71

После этого был осуществлён возврат в окно с рентгенограммой и расчёт соответствующего значения СЛН (см. рисунок 30).

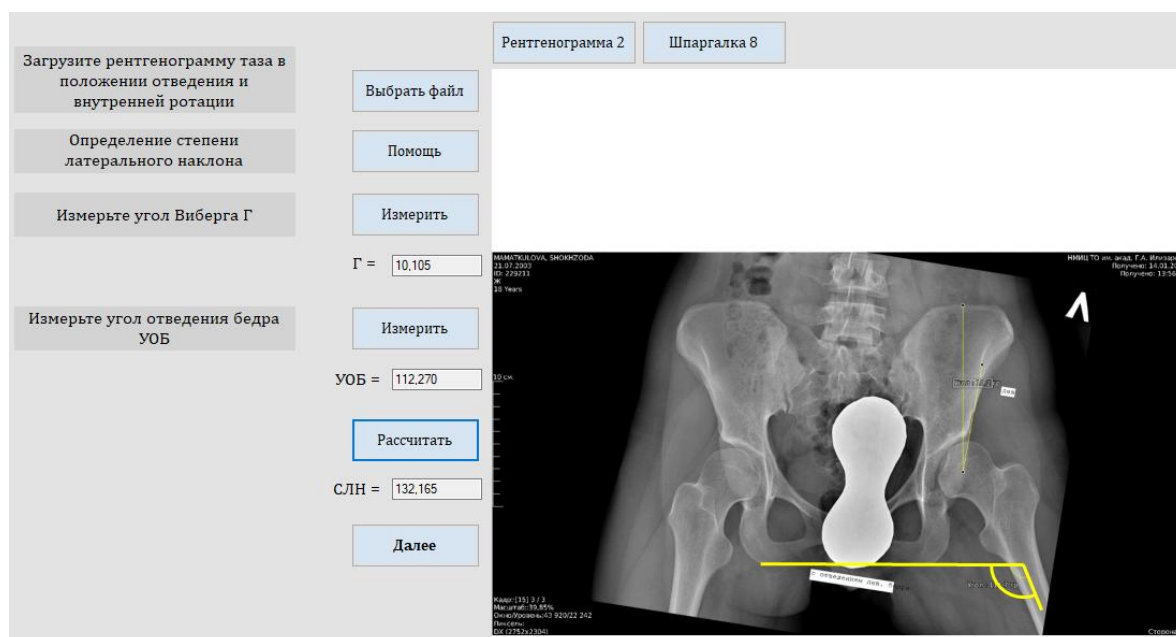


Рисунок 30 – Окно программы после измерений и расчёта СЛН.

Была загружена рентгенограмма 3, после чего открыта «шпаргалка» 9 (см. рисунок 31).

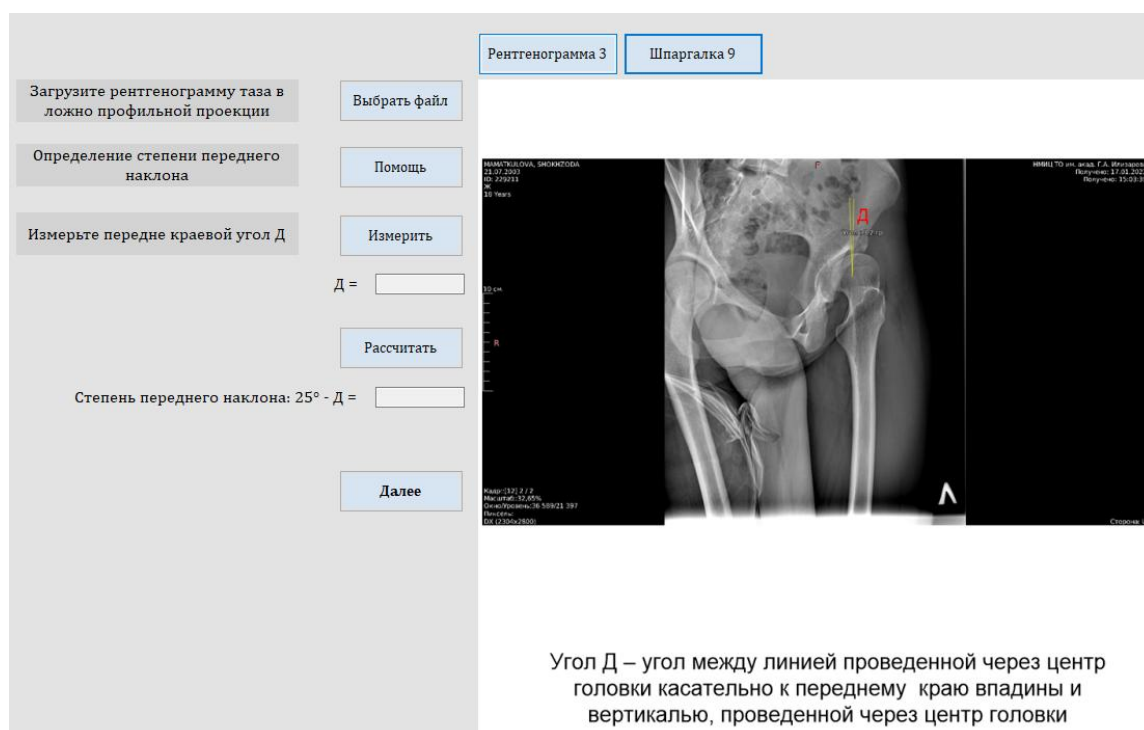


Рисунок 31 – Окно программы после открытия девятой шпаргалки

После чего был выполнен возврат в окно с рентгенограммой и измерение угла Д, расчёт степени переднего наклона и переход в следующее окно (см. рисунок 32).

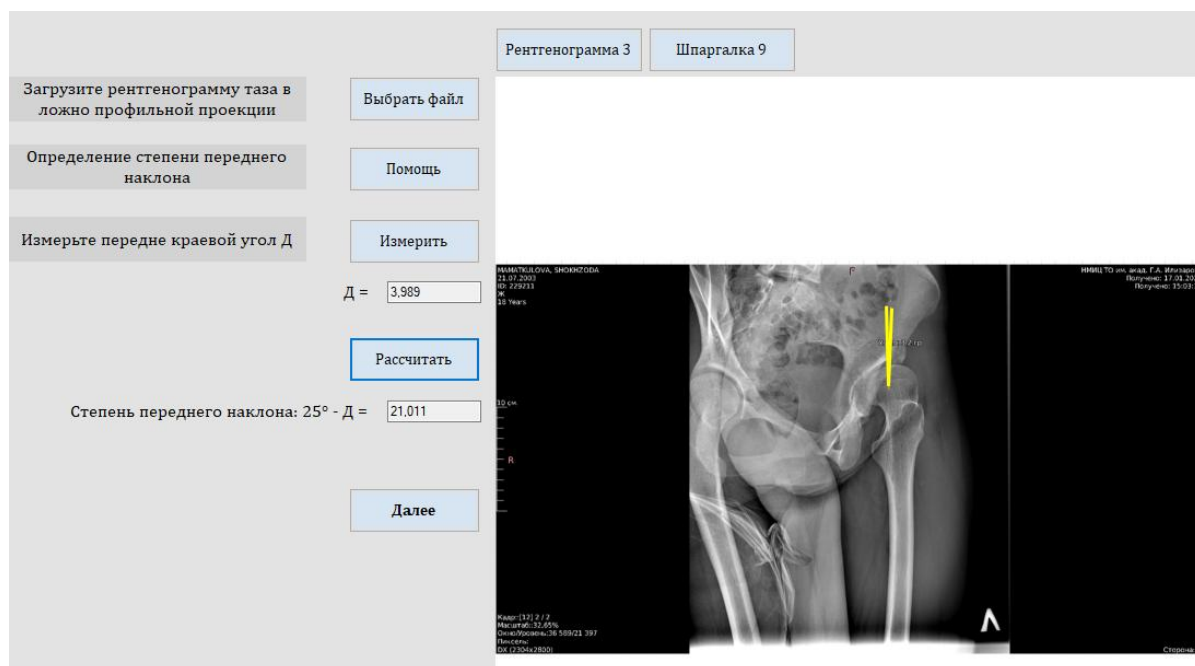


Рисунок 32 – Окно программы после измерений и расчёта степени переднего наклона.

Была открыта рентгенограмма 4, после чего был выполнен переход к «шпаргалке» 10 (см. рисунок 33).

Загрузите рентгенограмму таза в аксиальной проекции Выбрать файл

Определение степени переднего наклона Помощь

Измерьте угол E Измерить

E =

Рассчитать

Степень дополнительного латерального наклона =

Введите данные пациента

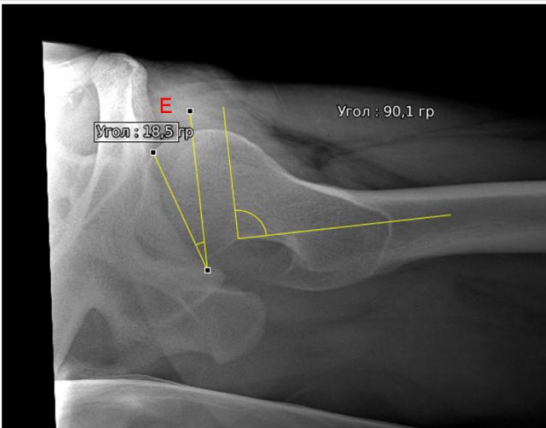
ФИО:

Год

Кол-во полных лет:

Сохранить вердикт в файл Завершить

Рентгенограмма 4
Шпаргалка 10



Определение степени переднего наклона

Определяется угол (E) между касательной к краям впадины и перпендикуляром к оси проксимального отдела диафиза бедра.

Степень переднего наклона - величина угла E.

Рисунок 33 – Окно программы после открытия десятой шпаргалки

После возврата к рентгенограмме (см. рисунок 34) были выполнены соответствующие измерения угла E, после чего выполнен расчёт степени дополнительного латерального наклона. Здесь же заполняются поля «ФИО:», «Год» и «Кол-во полных лет:».

Загрузите рентгенограмму таза в аксиальной проекции Выбрать файл

Определение степени переднего наклона Помощь

Измерьте угол E Измерить

E =

Рассчитать

Степень дополнительного латерального наклона =

Введите данные пациента

ФИО:

Год

Кол-во полных лет:

Сохранить вердикт в файл Завершить

Рентгенограмма 4
Шпаргалка 10

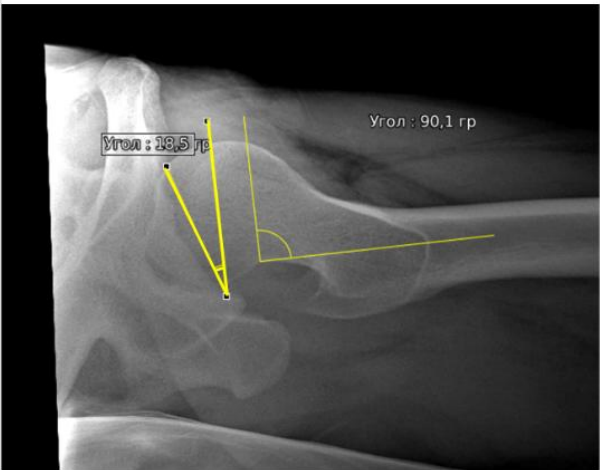


Рисунок 34 – Окно программы после измерений, расчёта и заполненными полями с данными пациента

Далее было выполнено нажатие на кнопку «Завершить», после чего появилось информирующее окно, показанное на рисунке 35.

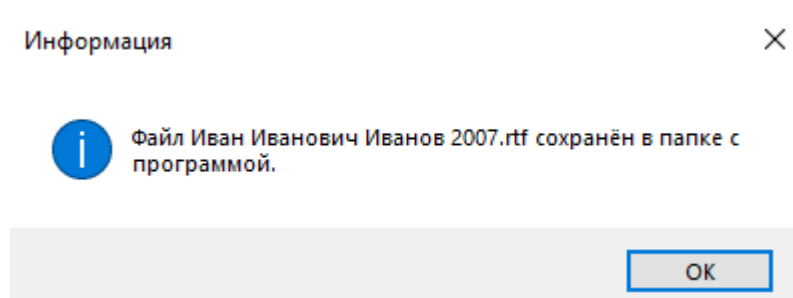


Рисунок 35 –Всплывающее окно после завершения формирования файла

После чего был открыт файл «Иван Иванович Иванов 2007.rtf» с вердиктом в папке с программой (см. рисунок 36).

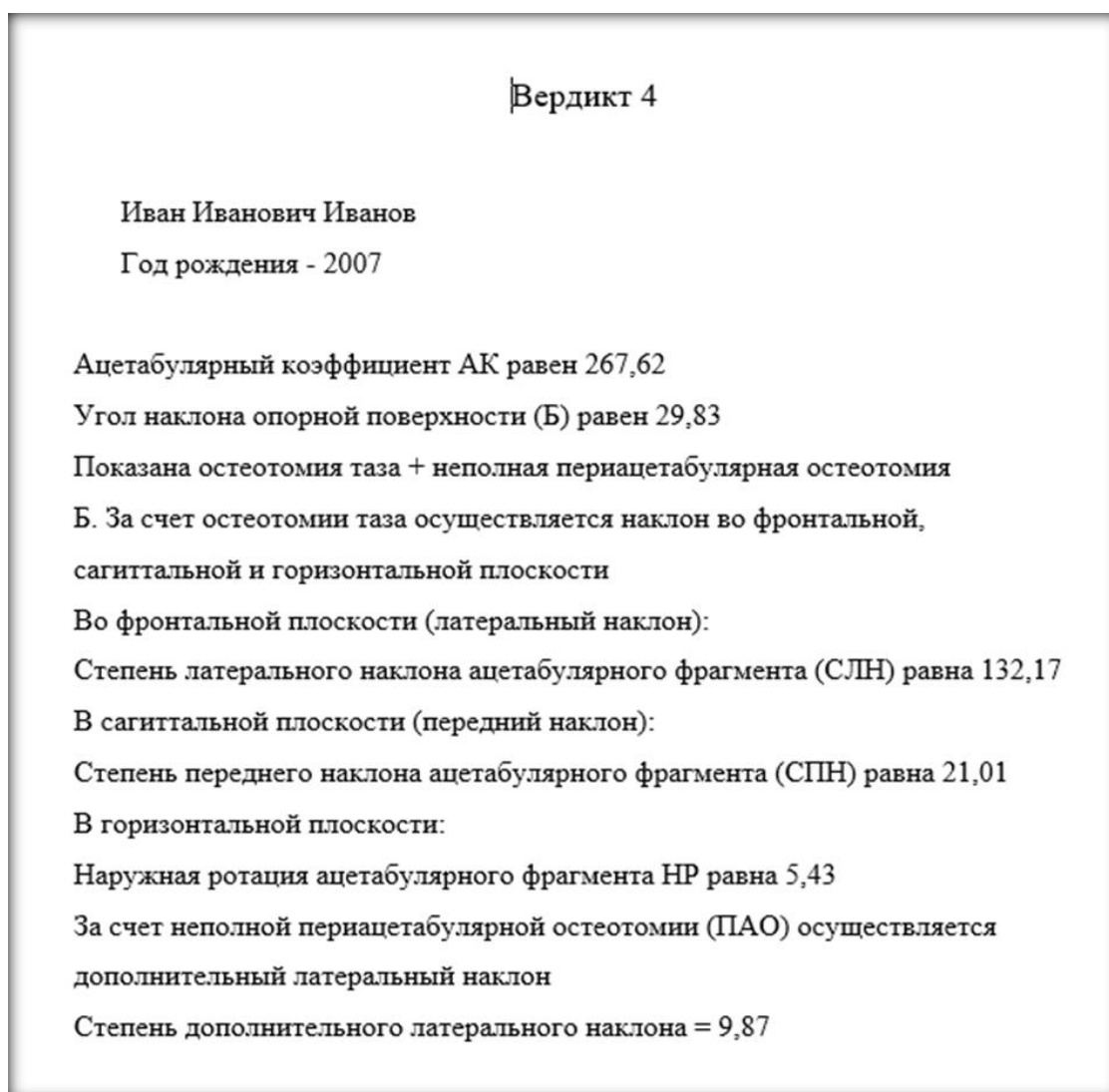


Рисунок 36 – Файл вердикта «Иван Иванович Иванов 2007.rtf»

Как видно из рисунка 36, программа успешно вывела результат в файл, следовательно, она работает исправно и в дальнейшем её можно применять для автоматизации обработки рентгеновских снимков.

					ПГУ1.090501.04.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

5 Расчет расходов на создание и внедрение программного обеспечения

5.1 Организация процесса разработки программы для автоматизации обработки рентгеновских снимков

Разработка программы проводилась удаленно с применением техники сотрудников, что позволило избежать материальных затрат на аренду офисного помещения, покупку и обслуживание техники.

По договоренности с персоналом ежемесячно им выплачивалась компенсация в размере 2000 рублей за использование личного оборудования в рабочих целях. Выплаты предоставлялись только постоянным сотрудникам.

Создание программы автоматизации обработки рентгеновских снимков выполнялась командой разработчиков, состав которой представлен в таблице 1.

Таблица 1 – Состав команды разработчиков

Специалист	Количество, чел.	Описание занятости
Руководитель проекта	1	– руководство проектом; – подбор команды исполнителей; – управление ресурсами проекта; – анализ предметной области; – контроль выполнения требований заказчика; – сдача и внедрение проекта
Программист	1	– написание и отладка программного обеспечения; – создание и утверждение программной документации; – сдача и внедрение программного кода.

Специалист	Количество, чел.	Описание обязанностей
Тестировщик	1	проведение тестирования программы;
Дизайнер	1	– проектирование интерфейса приложения; – разработка и оформление документации для пользователей.

Информация об основной заработной плате (ОЗП) штатных сотрудников и их численности представлена в таблице 2.

Таблица 2 – Данные о зарплате

Специалист	Количество, чел.	Общая зарплата, рублей
Руководитель проекта	1	70 000
Программист	1	50 000
Тестировщик	1	50 000

Сейчас обычно дизайнер не трудится в штате программистов, а с работодателем заключает договор ГПХ. Такой договор гражданско-правового типа предполагает выполнение работы или услуги не в качестве сотрудника, а как подрядчик. Договор ГПХ подписывается между работодателем и исполнителем.

Этот документ не предполагает включение исполнителя в штат предприятия, но подразумевает удержание налогов и страховых взносов с выплаченного вознаграждения.

Система налогообложения в данном случае соответствует налогообложению полученной заработной платы.

Информация о платежах по ГПХ приведена в таблице 3.

Таблица 3 – Платежи по ГПХ

Специалист	Число, человек	Общая сумма выплаты, рублей
Дизайнер	1	60 000

5.2 Определение себестоимости разработки программы для автоматизации обработки рентгеновских снимков

Себестоимость создания приложения охватывает следующие виды затрат:

- затраты на оплату труда (ЗП);
- расходы по договорам гражданско-правового характера (РГПХ);
- компенсационные выплаты сотрудникам (КОМП);
- расходы на сопровождение программы (ОП).

Расходы на оплату труда включают основную зарплату (ОЗП) сотрудников, участвующих в разработке программного обеспечения, и обязательные страховые взносы (ОСВ).

Аналогичный порядок применяется для расчетов по договорам ГПХ.

При пятидневной рабочей неделе с учётом выходных в среднем в месяце насчитывается 21 рабочий день.

На создание всего проекта выделяется три месяца.

5.2.1 Определение затрат на оплату труда разработчиков

На основании данных из таблицы 10, месячная заработная плата постоянных работников равна $70\,000 + 50\,000 + 50\,000 = 170\,000$ рублей.

Исходя из этого, затраты на оплату труда разработчиков за весь срок реализации проекта (ОЗП) составят:

$$\text{ОЗП} = 170\,000 \cdot 3 = 510\,000 \text{ (руб.)}$$

5.2.2 Расчет отчислений на страховые взносы

Компания, как разработчик программного обеспечения, обязана производить отчисления на страховые взносы с фонда оплаты труда (ФОТ). Эти отчисления направляются в Фонд пенсионного и социального страхования РФ и

регулируются Налоговым кодексом РФ (№ 239-ФЗ от 14.07.2022). Вместо стандартной ставки в 30%, компания имеет право применять льготную ставку 7,6% (подп. 3 п. 1 ст. 427 НК РФ) благодаря своей деятельности в сфере IT. Эта льгота распространяется на все выплаты сотрудникам, включая вознаграждения по договорам ГПХ (письмо Минфина от 23.12.2016 № 03-04-12/77644). Информация о затратах на оплату труда и начисленных страховых взносах представлена в таблице 4.

Таблица 4 - Структура расходов на заработную плату

Сотрудник	Заработная плата за месяц, рублей	Общая сумма отчислений за месяц, рублей	Всего за 1 месяц, рублей	Всего за 6 месяцев, рублей
Руководитель проекта	70 000	5 320	75 320	225 960
Программист	50 000	3 800	53 800	161 400
Тестировщик	50 000	3 800	53 800	161 400
Всего	170 000	12 920	86 080	548 760

В таблице приведена разбивка расходов на оплату труда по сотрудникам, включающая ежемесячную заработную плату, сумму отчислений и общую сумму выплат за месяц, а также общую сумму выплат за 3 месяца.

Таблица 5 содержит информацию о расходах на выплаты физическим лицам по договорам ГПХ и начисленным на эти выплаты страховым взносам.

Таблица 5 – Расходы на выплаты физическим лицам по договорам ГПХ

Сотрудник	Общая сумма выплаты, рублей	Общая сумма отчислений, рублей
Дизайнер	60 000	4 560

Фонд оплаты труда (ФОТ) команды разработчиков приложения, включая обязательные отчисления, составляет 548 760 рублей. Структура ФОТ включает в себя: заработную плату в размере 520 000 рублей и страховые взносы на сумму 77 760 рублей. Дополнительно предусмотрены компенсационные выплаты сотрудникам в размере 18 000 рублей:

$$2000 * 3 * 3 = 18\,000 \text{ руб.}$$

Себестоимость разработки определяется как совокупность всех затрат, представленных в таблице 6.

Таблица 6 – Совокупность всех затрат на реализацию проекта

Статья затрат	Сумма отчислений, рублей
Фонд оплаты труда, включая обязательные отчисления	548 760
Затраты на выплаты по договорам ГПХ, включая взносы	64 560
Дополнительно предусмотрены компенсационные выплаты сотрудникам	18 000
Итого:	631 320

Итоговая стоимость разработки проекта оценивается в 631 320 рублей, что соответствует его себестоимости.

6 Безопасность жизнедеятельности. Организация рабочего места с ПК

6.1 Требования к помещению для работы с ПК

Правильная организация рабочего места с компьютером – залог комфортной, безопасной и продуктивной работы. Эргономичное рабочее пространство снижает риск развития усталости, напряжения глаз и заболеваний опорно-двигательного аппарата.

Помещение должно быть достаточно просторным для размещения всего необходимого оборудования и обеспечения свободы движений. Рекомендуемая площадь – не менее 6-8 кв. м. Важно продумать расположение мебели и техники, чтобы минимизировать риск травм и обеспечить правильное положение тела во время работы.

Не менее важна эффективная система вентиляции, обеспечивающая постоянный приток свежего воздуха и поддержание комфортной температуры. Хорошая вентиляция предотвращает застой воздуха, снижая риск головных болей и утомляемости. Также необходимо контролировать уровень влажности, поддерживая его в оптимальном диапазоне для здоровья и работоспособности.

Для надежной и безопасной работы компьютерного оборудования необходимо обеспечить достаточное количество исправных розеток. Электропроводка должна быть в идеальном состоянии, без дефектов и перегрузок, чтобы предотвратить короткие замыкания и возгорания. Для защиты от перепадов напряжения и обеспечения бесперебойной работы рекомендуется использовать стабилизаторы напряжения и источники бесперебойного питания (ИБП), позволяющие корректно завершить работу в случае отключения электричества. Кроме того, для повышения концентрации и продуктивности необходимо минимизировать внешние шумы. Если рабочее место находится в

шумном месте, следует использовать звукоизоляцию, шумопоглощающие панели или наушники с шумоподавлением.

Организация искусственного освещения на рабочем месте должна обеспечивать равномерное распределение света по всей площади. Приоритет отдается использованию люминесцентных ламп, оснащенных рассеивателями и экранирующими решетками, которые позволяют получить мягкий и равномерный свет, минимизируя образование бликов и теней на экранах мониторов и рабочих поверхностях. Эксплуатация светильников без защитных элементов не рекомендуется, поскольку это может вызывать дискомфорт и повышенную нагрузку на зрение. Для организации локального освещения допускается применение ламп накаливания, включая галогенные, обеспечивающих более направленный и мягкий световой поток. Регулярное техническое обслуживание осветительных приборов и оконных проемов, включающее очистку от загрязнений и замену вышедших из строя ламп, является необходимым условием для поддержания требуемого уровня освещенности и создания благоприятной рабочей среды. В помещениях, оборудованных рабочими местами с компьютерами, должна быть предусмотрена система защитного заземления, обеспечивающая электробезопасность персонала в случае неисправности оборудования.

Санитарно-гигиенические требования включают регулярную влажную уборку для поддержания чистоты и снижения концентрации пыли и аллергенов, а также обязательное проветривание помещения после каждого часа работы для обеспечения притока свежего воздуха и предотвращения накопления углекислого газа. Для создания комфортных условий труда необходимо обеспечить эффективную работу систем отопления и кондиционирования, поддерживающих оптимальные параметры микроклимата: температуру в диапазоне 20-24°C и относительную влажность 40-60%. Соблюдение этих параметров способствует снижению утомляемости и повышению работоспособности сотрудников.

Помимо технических решений и правильной организации рабочего пространства, критически важно иметь под рукой аптечку для оказания первой помощи при травмах и внезапных недомоганиях. Для борьбы с возгораниями, особенно электрооборудования, необходим углекислотный огнетушитель. Раннее обнаружение пожара и предотвращение его распространения обеспечивается установкой пожарных извещателей.

В совокупности, эти меры не только создают безопасную и комфортную среду, но и положительно влияют на продуктивность, снижают риск профессиональных заболеваний и улучшают общее самочувствие пользователей компьютеров.

6.2 Требования к организации рабочих мест

Организация рабочих мест, использующих современные ЖК- или плазменные экраны, требует особого внимания к пространству и эргономике. Минимальная площадь на одно рабочее место должна составлять 4,5 кв. м, что обеспечивает достаточно места для комфортного размещения оборудования и передвижения сотрудника. Для предотвращения зрительного напряжения и физического дискомфорта необходимо соблюдать расстояние между столами с мониторами (не менее 2 метров) и между боковыми поверхностями экранов (не менее 1,2 метра). Рекомендуется располагать рабочие столы таким образом, чтобы естественный свет из окон падал на монитор сбоку (предпочтительно слева), что уменьшает блики и снижает нагрузку на глаза.

Для сотрудников, занятых сложной умственной работой, требующей концентрации и творческого подхода, рекомендуется использовать перегородки высотой 1,5-2 метра. Эти перегородки обеспечивают визуальную и акустическую изоляцию, снижая уровень отвлекающих факторов и способствуя сохранению концентрации.

Оптимальное расстояние от глаз пользователя до экрана монитора составляет 600-700 мм. Это расстояние обеспечивает комфортное восприятие

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		84

алфавитно-цифровой информации без излишнего напряжения зрительного аппарата. Минимальное допустимое расстояние до монитора – 500 мм.

Клавиатуру рекомендуется располагать на расстоянии 100-300 мм от переднего края стола, что способствует снижению нагрузки на руки и повышает удобство работы с текстом.

Современные рабочие столы, предназначенные для работы с компьютером, могут иметь различные конструктивные особенности, направленные на повышение эргономичности и удобства использования. Предпочтительным является использование столов, оснащенных выдвижной или отдельной площадкой для клавиатуры, что позволяет более эффективно организовать рабочее пространство. Регулируемая высота стола позволяет адаптировать рабочее место под индивидуальные антропометрические данные пользователя. В случае использования стола с фиксированной высотой, оптимальным является диапазон 680-800 мм.

Для поддержания здоровой осанки и комфорта на рабочем месте рекомендуется использовать регулируемые офисные кресла. Возможность настройки высоты сиденья, угла наклона спинки и сиденья позволяет адаптировать кресло под индивидуальные особенности пользователя, минимизируя риск возникновения болей в мышцах и суставах. Рабочее пространство должно быть организовано таким образом, чтобы все необходимые инструменты и оборудование, включая клавиатуру, мышь и документы, были легко доступны и не создавали ощущения стесненности. Важно, чтобы кресло обеспечивало надежную поддержку спины и предотвращало перенапряжение.

Эргономика рабочего места, предназначенного для работы в положении сидя, предполагает возможность регулировки взаимного расположения рабочей поверхности и сиденья. Это включает в себя настройку высоты и размеров стола, а также регулировку высоты и угла наклона подставки для ног (при ее использовании). Такая адаптация позволяет оптимизировать распределение нагрузки на опорно-двигательный аппарат и минимизировать риск развития

усталости. Кроме того, необходимо обеспечить возможность динамической смены позы в течение рабочего дня. Регулярная смена положения тела способствует улучшению кровообращения, снижению мышечного напряжения и, как следствие, повышению общей работоспособности и качества выполняемых задач.

					ПГУ1.090501.04.001 ПЗ	Лист
						86
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения проекта была разработана программа, предназначенная для автоматизации процессов сбора и обработки данных о состоянии опорно-двигательного аппарата человека путем анализа информации, получаемой при измерении параметров рентгеновских снимков.

Программа реализована на языке программирования C# с использованием технологии Windows Forms.

В процессе реализации проекта были сформулированы и выполнены основные этапы проектирования: разработаны алгоритмы работы программы, экранные формы интерфейса пользователя, осуществлена программная реализация разработки и ее тестирование, разработано руководство пользователя.

За счет внедрения и использования приложения для анализа и обработки рентгеновских снимков ожидаемый положительный эффект состоит в сокращении времени работы врачей (хирургов, травматологов, ортопедов) за счет автоматизации процессов сбора и обработки информации при работе с рентгеновскими снимками, определения диагноза и выдачи рекомендаций по хирургическому лечению пациента. Это, в свою очередь, приводит к снижению нагрузки на врача при обслуживании каждого пациента и позволяет увеличить количество принятых пациентов за сутки, что является особенно актуальным в настоящее время.

Также можно отметить возможность применения данного приложения в качестве обучающей программы в целях обучения молодых специалистов основам анализа и обработки рентгеновских снимков в рассматриваемой области медицины.

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		87

Список использованных источников

1. А.Н. Сенча А.Н., Д.В. Беляев Д.В. Ультразвуковая диагностика. Тазобедренный сустав. - М.: Издательский дом Видар#М, 2016. – 152 с.
2. Хорев П.Б. Объектно-ориентированное программирование с примерами на С#. – М: Издательство ФОРУМ, 2023, - 200 с.
3. Троелсен Э. С# и платформа .NET. Библиотека программиста. - Пер. с англ. - М.: Питер, 2002. - 800 с.
4. Андриевская Н.К., Мартыненко Т.В., Турупалов В. В. Основы визуального программирования в среде Visual Studio на базе С#. – М.: Инфра-Инженерия, 2023, - 232 с.
5. Джонсон Б. Основы Microsoft Visual Studio .NET 2003. - Пер. с англ. - М.: Русская Редакция, 2022. - 773 с.
6. Понамарев В.П. Программирование на C++/C# в Visual Studio .NET 2003. - М.: БХВ-Петербург, 2021. - 322 с.
7. Мурадханов С. Э. Разработка на языке С# приложений с графическим интерфейсом: использование Windows Forms. – М.: Издательский Дом НИТУ «МИСиС»: 2019, - 396 с.
8. Натан А. WPF 4. Подробное руководство. - Пер. с англ. - СПб.: Символ-Плюс, 2011. - 880 с.

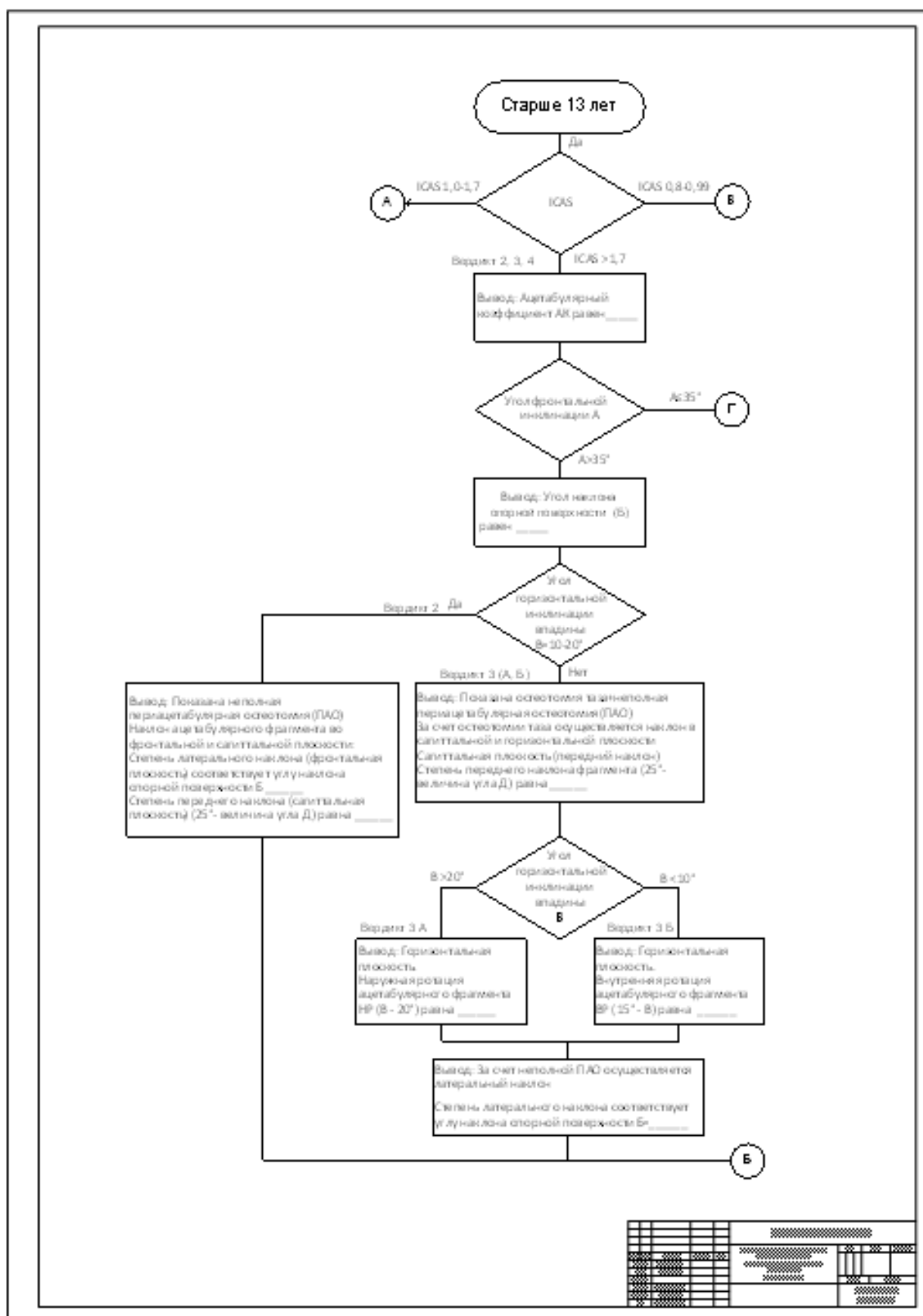
Приложение А

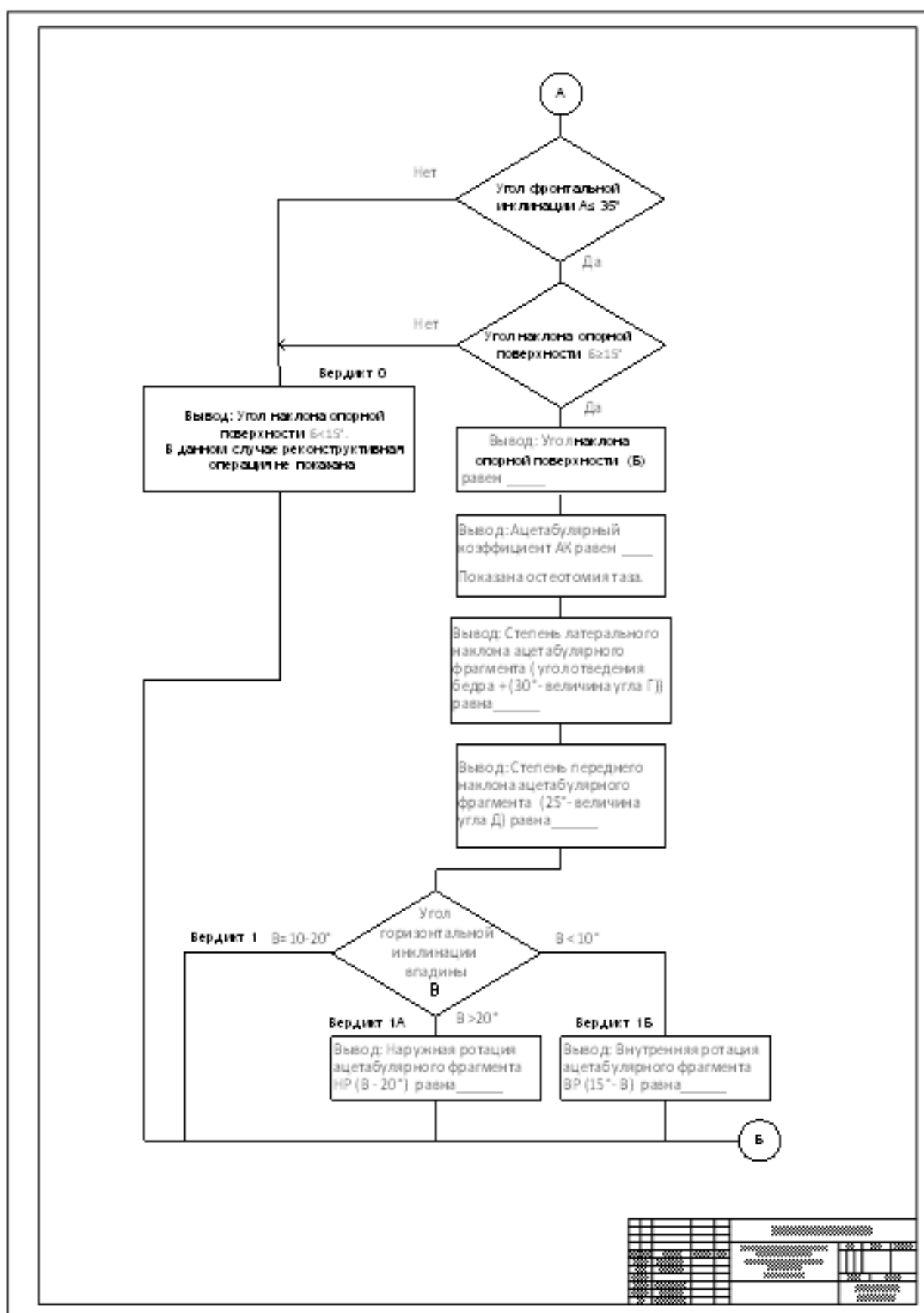
(обязательное)

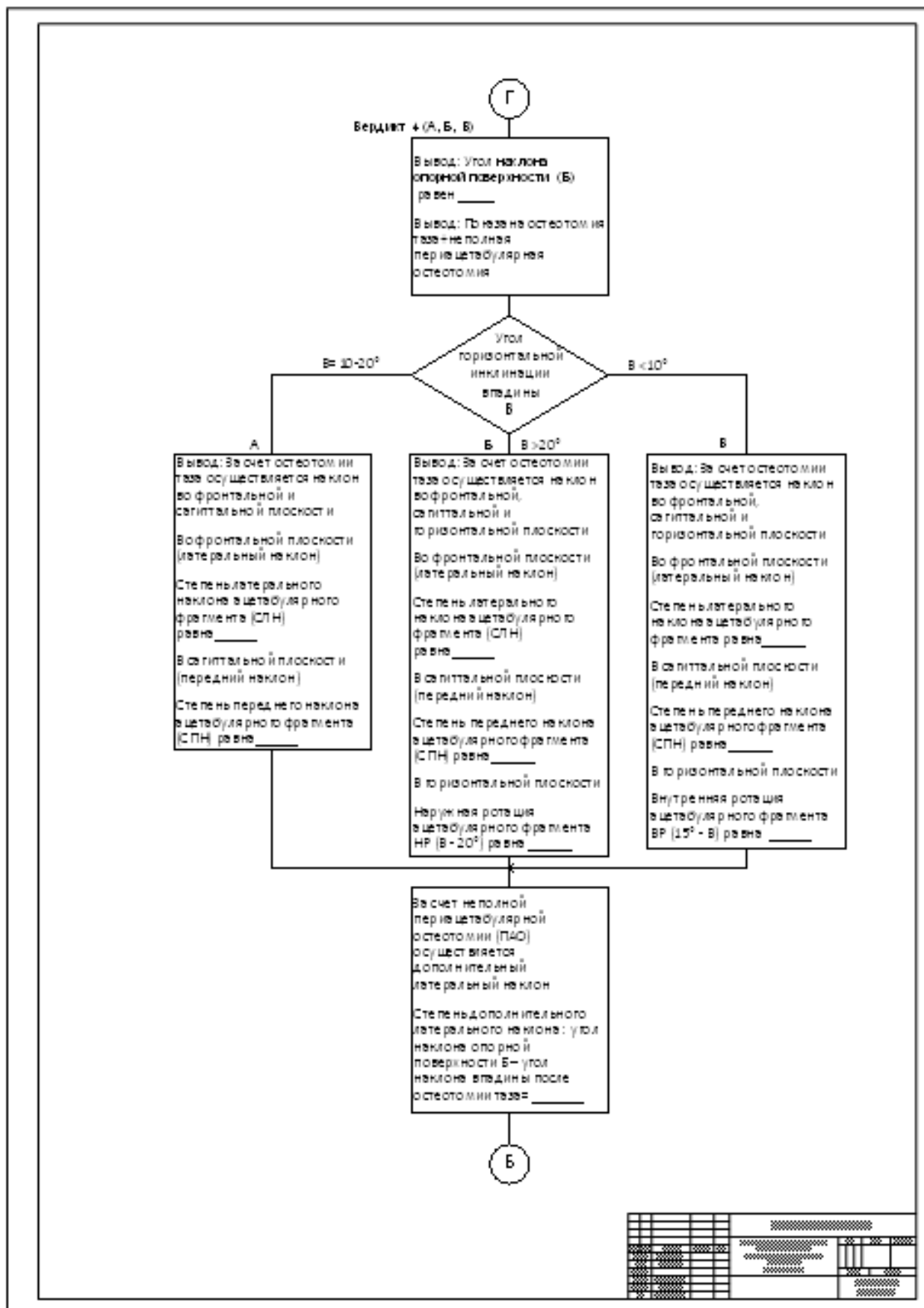
Программа автоматизации обработки рентгеновских снимков

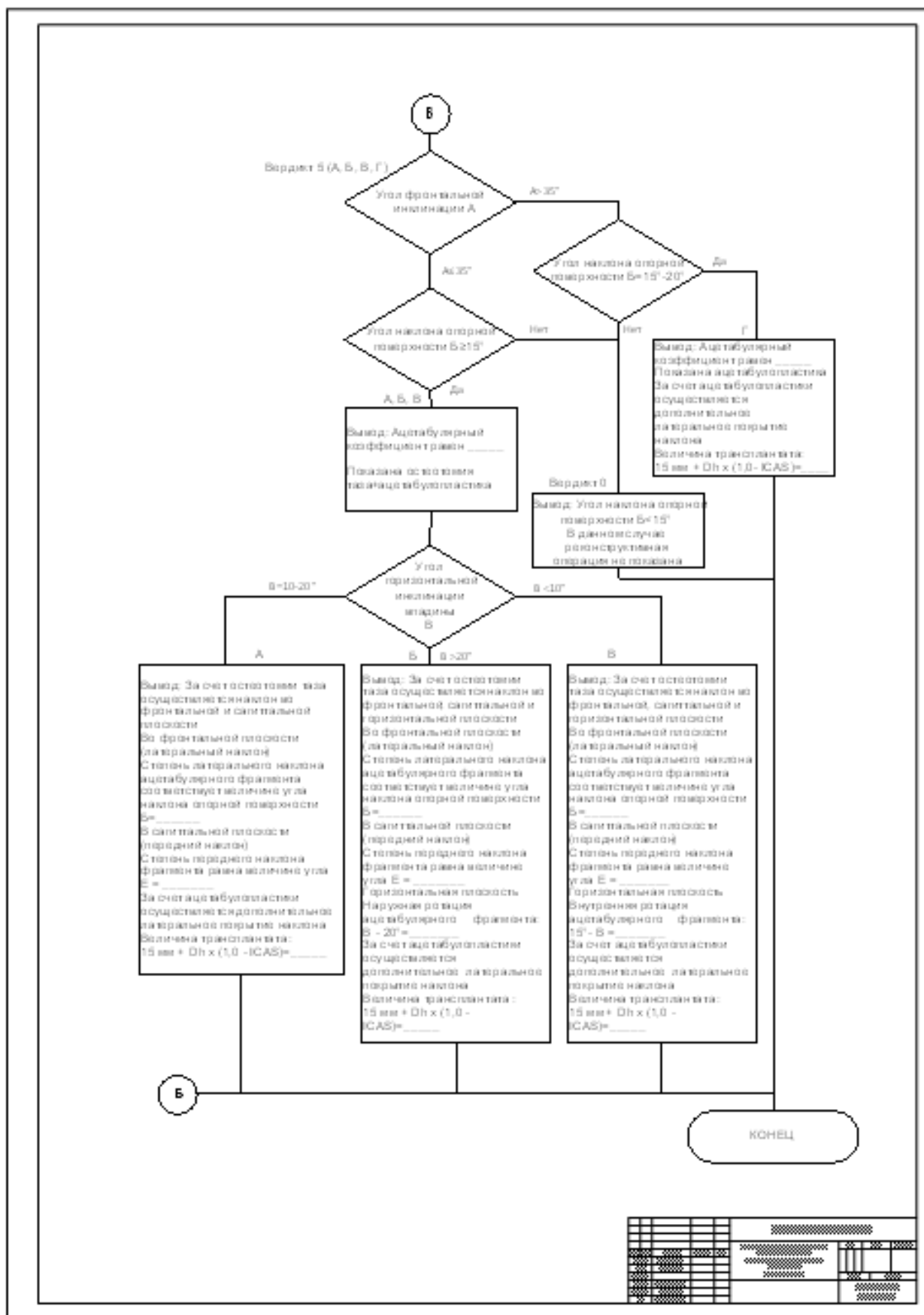
Схемы алгоритмов

					ПГУ1.090501.04.001 ПЗ	Лист
						89
Изм.	Лист	№ докум.	Подпись	Дата		









Приложение Б

(обязательное)

Программа автоматизации обработки рентгеновских снимков

Листинг программы

					ПГУ1.090501.04.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		98

Файл Program.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms; namespace Medic
{
    internal static class Program
    {
        /// Главная точка входа для приложения.
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
Application.ApplicationExit += OnApplicationExit;
Application.Run(new Window_1());
        }
        private static void OnApplicationExit(object sender, EventArgs
e)
        {
            SharedData.DisposeSharedPic1();
            SharedData.DisposeSharedPic2();
            SharedData.DisposeSharedPic3();
            SharedData.DisposeSharedPic4();
        }
    }
}

```

Файл SharedData.cse

```

using System.Drawing; namespace Medic
{
    public static class SharedData
    {
        // Поля для хранения загруженных пользователем рентгенограмм
        public static Bitmap Radiograph_1 { get; set; }
        public static Bitmap Radiograph_2 { get; set; }
        public static Bitmap Radiograph_3 { get; set; }
        public static Bitmap Radiograph_4 { get; set; } // Поля
для хранения всех данных, участвующих при вынесении вердикта
        public static double Dh;
        public static double Da;
        public static double R;
        public static double ISh;
        public static double Dist_A;
        public static double Dist_B;
        public static double Dist_D;
        public static double Dist_W;
        public static double Okano;
        public static double Angle_A;
        public static double Angle_B;
        public static double Angle_V;
        public static double Angle_Viberg;
        public static double Angle_D;
        public static double Angle_E;
        public static double ISA;
    }
}

```

```

        public static double ICAS;
        public static double AK;
        public static double UOB;
        public static double SLN;
        public static double SPN_1;
        public static double SDLN;
        public static double HP;
        public static double BP;
        public static double trans;
        public static string PHIO;
        public static int year;
        public static int age; // Методы для освобождения
ресурсов, когда изображение больше не нужно
        // (например, при закрытии приложения)
        public static void DisposeSharedPic1()
        {
            Radiograph_1?.Dispose();
            Radiograph_1 = null;
        }
        public static void DisposeSharedPic2()
        {
            Radiograph_2?.Dispose();
            Radiograph_2 = null;
        }
        public static void DisposeSharedPic3()
        {
            Radiograph_3?.Dispose();
            Radiograph_3 = null;
        }
        public static void DisposeSharedPic4()
        {
            Radiograph_4?.Dispose();
            Radiograph_4 = null;
        }
    }
}

```

Файл Window_1.cs

```

using System;
using System.Drawing;
using System.IO;
using System.Windows.Forms; namespace Medic
{
    public partial class Window_1 : Form
    {
        private Image clueImage_1;
        private bool isDrawingMode = false;
        private bool isDrawingCircleMode = false;
        private bool isFirstPointSet = false;
        private Point firstPoint;
        private Point currentMousePosition;
        private Bitmap drawingLayer;
        public Window_1()
        {
            InitializeComponent();
            LoadClueImage();
            btn_switch_1.Enabled = false;
            btn_switch_2.Enabled = false;
            label_1_3.Visible = false;
            pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;
        }
    }
}

```

```

        pictureBox1.MouseDown += PictureBox1_MouseDown;
        pictureBox1.MouseMove += PictureBox1_MouseMove;
        pictureBox1.Paint += PictureBox1_Paint; //
Изначально кнопка неактивна
        btn_measure_1_2.Enabled = true;
        btn_measure_1_1.Enabled = true;
    }
    private void LoadClueImage()
    {
        try
        {
            string clueImagePath =
Path.Combine(Application.StartupPath, "clue_1.png"); if
(File.Exists(clueImagePath))
            {
                clueImage_1 = Image.FromFile(clueImagePath);
            }
            else
            {
                MessageBox.Show("Файл clue_1.png не найден в папке
с программой!",
                                "Предупреждение",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Warning);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка при загрузке clue_1.png:
{ex.Message}",
                                "Ошибка",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Error);
        }
    }
    private void btn_next_1_Click(object sender,
EventArgs e)
    {
        if (textBox_1_3_Ish.Text == "")
        {
            MessageBox.Show("Индекс Ish не измерен!",
                                "Ошибка",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Error);
        }
        else
        {
            if (SharedData.Ish >= 0.6 && SharedData.Ish <= 1.0)
            {
                Window_2 w2 = new Window_2();
                w2.Show();
                Hide();
            }
            else
            {
                MessageBox.Show("Индекс сферичности головки Ish
выходит за пределы требуемого диапазона.\r\nПрименение программы
невозможно",

```

```

        "Ошибка",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
private void btn_select_1_Click(object
sender, EventArgs e)
{
    using (OpenFileDialog openFileDialog = new
OpenFileDialog())
    {
        openFileDialog.Title = "Выберите изображение";
        openFileDialog.Filter = "Файлы
изображений|*.png;*.jpg;*.jpeg|PNG файлы|*.png|JPEG
файлы|*.jpg;*.jpeg";
        if (openFileDialog.ShowDialog() ==
DialogResult.OK)
        {
            try
            {
                SharedData.Radiograph_1 =
(Bitmap) Image.FromFile(openFileDialog.FileName);
                pictureBox1.Image = SharedData.Radiograph_1;
                btn_switch_1.Enabled = true;
                btn_switch_2.Enabled = (clueImage_1 != null);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка при загрузке
изображения: {ex.Message}",
                                "Ошибка",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Error);
            }
        }
    }
}
private void btn_switch_1_Click(object sender,
EventArgs e)
{
    if (SharedData.Radiograph_1 != null)
    {
        pictureBox1.Image = SharedData.Radiograph_1;
    }
}
private void btn_switch_2_Click(object sender,
EventArgs e)
{
    if (clueImage_1 != null)
    {
        pictureBox1.Image = clueImage_1;
        drawingLayer?.Dispose();
        drawingLayer = null;
        pictureBox1.Invalidate();
    }
    else
    {
        MessageBox.Show("Файл - шпаргалка clue_1.png не
найден!",
                        "Ошибка",

```

```

        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
private void btn_help_1_Click(object sender,
EventArgs e)
{
    label_1_3.Visible = !label_1_3.Visible;
}
private void btn_measure_1_1_Click(object sender,
EventArgs e)
{
    if (SharedData.Radiograph_1 != null)
    {
        if (pictureBox1.Image != SharedData.Radiograph_1)
        {
            MessageBox.Show("Рентгенограммы 1 нет на экране!",
                "Ошибка",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        else
        {
            // Сброс состояния и включение режима рисования
            isDrawingMode = true;
            isFirstPointSet = false;
            textBox_1_2_R.Text = "0"; //
Очистка предыдущего рисунка
            if (drawingLayer != null)
            {
                drawingLayer.Dispose();
                drawingLayer = null;
            } // Перерисовка PictureBox
            pictureBox1.Invalidate();
        }
    }
    else
    {
        MessageBox.Show("Сначала загрузите рентгенограмму!",
            "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
}
private void
PictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    if (!isDrawingMode || e.Button != MouseButtons.Left)
        return; // Получение координат относительно
изображения
    Point imagePoint = GetImageCoordinates(e.Location);
    if (!isFirstPointSet)
    {
        // Установка первой точки
        firstPoint = imagePoint;
        isFirstPointSet = true;
        currentMousePosition = imagePoint;
    }
    else
    {

```

```

        // Установка второй точки и завершение рисования
        if (isDrawingCircleMode)
            CreateFinalCircle(imagePoint);
        else
            CreateFinalLine(imagePoint);
isDrawingMode = false;
isDrawingCircleMode = false;
isFirstPointSet = false;
    }
} private void PictureBox1_MouseMove(object sender,
MouseEventArgs e)
{
    if (!isDrawingMode || !isFirstPointSet)
        return; // Обновление текущей позиции мыши
    currentMousePosition = GetImageCoordinates(e.Location);
// Расчёт и отображение длины/диаметра
    double length = CalculateDistance(firstPoint,
currentMousePosition); if (isDrawingCircleMode)
        textBox_1_1_Dh.Text = Math.Round(length,
2).ToString();
    else
        textBox_1_2_R.Text = Math.Round(length, 2).ToString();
// Перерисовка для отображения временной фигуры
    pictureBox1.Invalidate();
} private void PictureBox1_Paint(object sender,
PaintEventArgs e)
{
    // Отрисовка сохранённой фигуры
    if (drawingLayer != null && pictureBox1.Image != null)
    {
        // Получение коэффициентов масштабирования
        float scaleX = (float)pictureBox1.Width /
pictureBox1.Image.Width;
        float scaleY = (float)pictureBox1.Height /
pictureBox1.Image.Height;
        float scale = Math.Min(scaleX, scaleY);
// Вычисление смещения для центрирования
        int offsetX = (int)((pictureBox1.Width -
pictureBox1.Image.Width * scale) / 2);
        int offsetY = (int)((pictureBox1.Height -
pictureBox1.Image.Height * scale) / 2); // Отрисовка
        // слоя с фигурой
        e.Graphics.DrawImage(drawingLayer, offsetX, offsetY,
pictureBox1.Image.Width * scale,
pictureBox1.Image.Height * scale);
    } // Отрисовка временной фигуры при рисовании
    if (isDrawingMode && isFirstPointSet)
    {
        using (Pen pen = new Pen(Color.Red, 2))
        {
            Point displayFirst =
GetDisplayCoordinates(firstPoint);
            Point displayCurrent =
GetDisplayCoordinates(currentMousePosition); if
(isDrawingCircleMode)
            {

```

```

        // Рисование окружности
        DrawCircleByDiameter(e.Graphics, pen,
displayFirst, displayCurrent);
    }
    else
    {
        // Рисование линии
        e.Graphics.DrawLine(pen, displayFirst,
displayCurrent);
        // Отрисовка точек
        e.Graphics.FillEllipse(Brushes.Red,
displayFirst.X - 3, displayFirst.Y - 3, 6, 6);
        e.Graphics.FillEllipse(Brushes.Red,
displayCurrent.X - 3, displayCurrent.Y - 3, 6, 6);
    }
}
}
}
private void CreateFinalLine(Point secondPoint)
{
    if (pictureBox1.Image == null)
        return;
    // Создание слоя для рисования
    drawingLayer = new Bitmap(pictureBox1.Image.Width,
pictureBox1.Image.Height);
    using (Graphics g =
Graphics.FromImage(drawingLayer))
    {
        g.Clear(Color.Transparent);
        g.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
        using
        (Pen pen = new Pen(Color.Red, 2))
        {
            g.DrawLine(pen, firstPoint, secondPoint);
            // Отрисовка точек
            g.FillEllipse(Brushes.Red, firstPoint.X - 3,
firstPoint.Y - 3, 6, 6);
            g.FillEllipse(Brushes.Red, secondPoint.X - 3,
secondPoint.Y - 3, 6, 6);
        }
        // Расчёт финальной длины
        double length = CalculateDistance(firstPoint,
secondPoint);
        textBox_1_2_R.Text = Math.Round(length, 2).ToString();
        pictureBox1.Invalidate();
    }
    private void CreateFinalCircle(Point secondPoint)
    {
        if (pictureBox1.Image == null)
            return;
        // Создание слоя для рисования
        drawingLayer = new Bitmap(pictureBox1.Image.Width,
pictureBox1.Image.Height);
        using (Graphics g =
Graphics.FromImage(drawingLayer))
        {
            g.Clear(Color.Transparent);
            g.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;
            using
            (Pen pen = new Pen(Color.Red, 2))
            {
                // Рисование окружности на слое изображения

```

```

        DrawCircleByDiameterOnImage(g, pen, firstPoint,
secondPoint);
    }
    } // Расчёт финального диаметра
    double diameter = CalculateDistance(firstPoint,
secondPoint);
    textBox_1_1_Dh.Text = Math.Round(diameter, 2).ToString();
pictureBox1.Invalidate();
} private void DrawCircleByDiameter(Graphics g, Pen
pen, Point p1, Point p2)
{
    // Вычисление центра окружности (середина диаметра)
    int centerX = (p1.X + p2.X) / 2;
    int centerY = (p1.Y + p2.Y) / 2; // Вычисление
радиуса
    double diameter = CalculateDistance(p1, p2);
    float radius = (float)(diameter / 2); //
Рисование окружности
    if (radius > 0)
    {
        g.DrawEllipse(pen, centerX - radius, centerY - radius,
radius * 2, radius * 2);
    } // Рисование диаметра
    g.DrawLine(pen, p1, p2); // Отрисовка точек на
концах диаметра
    g.FillEllipse(Brushes.Red, p1.X - 3, p1.Y - 3, 6, 6);
    g.FillEllipse(Brushes.Red, p2.X - 3, p2.Y - 3, 6, 6);
    // Отрисовка центра окружности
    using (Brush centerBrush = new SolidBrush(Color.Blue))
    {
        g.FillEllipse(centerBrush, centerX - 2, centerY - 2,
4, 4);
    }
} private void DrawCircleByDiameterOnImage(Graphics g,
Pen pen, Point p1, Point p2)
{
    // Вычисление центра окружности (середина диаметра)
    int centerX = (p1.X + p2.X) / 2;
    int centerY = (p1.Y + p2.Y) / 2; // Вычисление
радиуса
    double diameter = CalculateDistance(p1, p2);
    float radius = (float)(diameter / 2); //
Рисование окружности
    if (radius > 0)
    {
        g.DrawEllipse(pen, centerX - radius, centerY - radius,
radius * 2, radius * 2);
    } // Рисование диаметра
    g.DrawLine(pen, p1, p2); // Отрисовка точек на
концах диаметра
    g.FillEllipse(Brushes.Red, p1.X - 3, p1.Y - 3, 6, 6);
    g.FillEllipse(Brushes.Red, p2.X - 3, p2.Y - 3, 6, 6);
    // Отрисовка центра окружности
    using (Brush centerBrush = new SolidBrush(Color.Blue))
    {

```

```

        g.FillEllipse(centerBrush, centerX - 2, centerY - 2,
4, 4);
    }
}
private Point GetImageCoordinates(Point
pictureBoxPoint)
{
    if (pictureBox1.Image == null)
        return Point.Empty; // Получение размеров и
позиции изображения в PictureBox
        float scaleX = (float)pictureBox1.Width /
pictureBox1.Image.Width;
        float scaleY = (float)pictureBox1.Height /
pictureBox1.Image.Height;
        float scale = Math.Min(scaleX, scaleY); int
offsetX = (int)((pictureBox1.Width - pictureBox1.Image.Width * scale)
/ 2);
        int offsetY = (int)((pictureBox1.Height -
pictureBox1.Image.Height * scale) / 2); // Преобразование
координат
        int imageX = (int)((pictureBoxPoint.X - offsetX) / scale);
        int imageY = (int)((pictureBoxPoint.Y - offsetY) / scale);
// Ограничение координат размерами изображения
        imageX = Math.Max(0, Math.Min(imageX,
pictureBox1.Image.Width - 1));
        imageY = Math.Max(0, Math.Min(imageY,
pictureBox1.Image.Height - 1)); return new Point(imageX,
imageY);
}
private Point GetDisplayCoordinates(Point imagePoint)
{
    if (pictureBox1.Image == null)
        return Point.Empty; float scaleX =
(float)pictureBox1.Width / pictureBox1.Image.Width;
        float scaleY = (float)pictureBox1.Height /
pictureBox1.Image.Height;
        float scale = Math.Min(scaleX, scaleY); int
offsetX = (int)((pictureBox1.Width - pictureBox1.Image.Width * scale)
/ 2);
        int offsetY = (int)((pictureBox1.Height -
pictureBox1.Image.Height * scale) / 2); int displayX =
(int)(imagePoint.X * scale + offsetX);
        int displayY = (int)(imagePoint.Y * scale + offsetY);
return new Point(displayX, displayY);
}
private double CalculateDistance(Point p1, Point p2)
{
    double dx = p2.X - p1.X;
    double dy = p2.Y - p1.Y;
    return Math.Sqrt(dx * dx + dy * dy);
}
private void btn_measure_1_2_Click(object sender,
EventArgs e)
{
    if (SharedData.Radiograph_1 != null)
    {
        if (pictureBox1.Image != SharedData.Radiograph_1)
        {
            MessageBox.Show("Рентгенограммы 1 нет на экране!",
"Ошибка",

```

```

        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
    else
    {
        // Сброс состояния и включение режима рисования
        isDrawingMode = true;
        isDrawingCircleMode = true;
        isFirstPointSet = false;
        textBox_1_1_Dh.Text = "0";
        Очистка предыдущего рисунка
        if (drawingLayer != null)
        {
            drawingLayer.Dispose();
            drawingLayer = null;
        }
        // Перерисовка PictureBox
        pictureBox1.Invalidate();
    }
}
else
{
    MessageBox.Show("Сначала загрузите рентгенограмму!",
        "Ошибка",
        MessageBoxButtons.OK,
        MessageBoxIcon.Error);
}
}
private void btn_measure_1_3_Click(object sender,
EventArgs e)
{
    if (textBox_1_1_Dh.Text == "" || textBox_1_2_R.Text == "")
    {
        MessageBox.Show("Требуемые значения не измерены!",
            "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
    else
    {
        SharedData.Dh = Convert.ToDouble(textBox_1_1_Dh.Text);
        SharedData.R = Convert.ToDouble(textBox_1_2_R.Text);
        SharedData.ISh = (SharedData.Dh / (SharedData.R *
0.5));
        textBox_1_3_ISh.Text =
Convert.ToString((float)SharedData.ISh);
    }
}
}
}

```

Файл Window_3.cs

```

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Windows.Forms; namespace Medic

```

```

{
    public partial class Window_3 : Form
    {
        private Image clueImage_3;
        private Image clueImage_4;          private enum DrawingPhase
        {
            None,                // Ничего не рисуется
            WaitingForP1,        // Ожидание первой точки
            WaitingForP2,        // Ожидание второй точки (первая линия
динамическая)
            WaitingForP3,        // Ожидание третьей точки (вторая линия и
угол динамические)
            DrawingCompleted // Все три точки установлены
        }
        private DrawingPhase currentPhase =
DrawingPhase.None;
        private List<PointF> drawnPoints = new List<PointF>();
        private PointF currentCursorPoint; // Текущая позиция курсора
для динамического рисования
        private TextBox activeAngleTextBox = null; // TextBox для
вывода текущего угла
        private Pen drawingPen; // Перо для рисования          private
const float ARC_RADIUS = 30f; // Радиус дуги для отображения угла
        public Window_3()
        {
            InitializeComponent();
            LoadClueImage();
            pictureBox3.Image = SharedData.Radiograph_1;
            label_3_4.Visible = false;
            label_3_5.Visible = false;
            DoubleBuffered = true;
            drawingPen = new Pen(Color.Yellow, 3);
        }
        private void btn_switch_3_1_Click(object sender,
EventArgs e)
        {
            pictureBox3.Image = SharedData.Radiograph_1;
        }
        private void btn_switch_3_2_Click(object sender,
EventArgs e)
        {
            pictureBox3.Image = clueImage_3;
            drawnPoints.Clear();
            pictureBox3.Invalidate();
        }
        private void btn_switch_3_3_Click(object sender,
EventArgs e)
        {
            pictureBox3.Image = clueImage_4;
            drawnPoints.Clear();
            pictureBox3.Invalidate();
        }
        private void LoadClueImage()
        {
            try
            {
                string clue3ImagePath =
Path.Combine(Application.StartupPath, "clue_3.png");
                string clue4ImagePath =
Path.Combine(Application.StartupPath, "clue_4.png");
                if
(File.Exists(clue3ImagePath) && File.Exists(clue4ImagePath))

```

```

        {
            clueImage_3 = Image.FromFile(clue3ImagePath);
            clueImage_4 = Image.FromFile(clue4ImagePath);
        }
        else
        {
            MessageBox.Show("Файлы clue_3.png или clue_4.png
не найдены в папке с программой!",
                            "Предупреждение",
                            MessageBoxButtons.OK,
                            MessageBoxIcon.Warning);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при загрузке clue_3.png или
clue_4.png: {ex.Message}",
                        "Ошибка",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
}

private void btn_help_3_Click(object sender,
EventArgs e)
{
    label_3_4.Visible = !label_3_4.Visible;
    label_3_5.Visible = !label_3_5.Visible;
}

private void btn_measure_3_1_Click(object sender,
EventArgs e)
{
    if (pictureBox3.Image != SharedData.Radiograph_1)
    {
        MessageBox.Show("Рентгенограммы 1 нет на экране!",
                        "Ошибка",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
    else StartMeasurement(textBox_3_1_Angle_A);
}

private void btn_measure_3_2_Click(object sender,
EventArgs e)
{
    if (pictureBox3.Image != SharedData.Radiograph_1)
    {
        MessageBox.Show("Рентгенограммы 1 нет на экране!",
                        "Ошибка",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
    }
    else StartMeasurement(textBox_3_2_Angle_B);
}

// Общий метод для начала/сброса измерения
private void StartMeasurement(TextBox targetTextBox)
{
    currentPhase = DrawingPhase.WaitingForP1;
    drawnPoints.Clear();
    activeAngleTextBox = targetTextBox;
    if (activeAngleTextBox != null)
    {

```

```

        activeAngleTextBox.Text = ""; // Очищаем поле угла
    }
    pictureBox3.Invalidate(); // Перерисовать PictureBox
(очистит предыдущие линии)
    // Убедимся, что PictureBox может получать фокус для
событий мыши, если нужно
    // pictureBox3.Focus(); // Обычно не требуется для
PictureBox
    } // Событие нажатия кнопки мыши на PictureBox
    private void pictureBox3_MouseDown(object sender,
MouseEventArgs e)
    {
        if (activeAngleTextBox == null || e.Button !=
MouseButtons.Left)
            return;
        switch (currentPhase)
        {
            case DrawingPhase.WaitingForP1:
                drawnPoints.Add(e.Location);
                currentPhase = DrawingPhase.WaitingForP2;
                currentCursorPoint = e.Location; // Инициализируем
для первого кадра
                break;
            case DrawingPhase.WaitingForP2:
                drawnPoints.Add(e.Location);
                currentPhase = DrawingPhase.WaitingForP3;
                currentCursorPoint = e.Location; // Инициализируем
для первого кадра
                break;
            case DrawingPhase.WaitingForP3:
                drawnPoints.Add(e.Location);
                currentPhase = DrawingPhase.DrawingCompleted;
                UpdateAngleDisplay(true); // Обновить угол в
TextBox окончательно
                break;
        }
        pictureBox3.Invalidate(); // Запросить перерисовку
    } // Событие перемещения мыши над PictureBox
    private void pictureBox3_MouseMove(object sender,
MouseEventArgs e)
    {
        if (activeAngleTextBox == null)
            return;
        currentCursorPoint = e.Location;
        if (currentPhase == DrawingPhase.WaitingForP2 || currentPhase ==
DrawingPhase.WaitingForP3)
        {
            if (currentPhase == DrawingPhase.WaitingForP3 &&
drawnPoints.Count == 2)
            {
                UpdateAngleDisplay(false); // Динамическое
обновление угла в TextBox
            }
            pictureBox3.Invalidate(); // Запросить перерисовку для
динамического отображения
        }
    } // Событие перерисовки PictureBox

```

```

private void pictureBox3_Paint(object sender, PaintEventArgs
e)
{
    // Устанавливаем высокое качество рендеринга
    e.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
    if (activeAngleTextBox == null && currentPhase == DrawingPhase.None)
    {
        // Если не в режиме измерения, ничего не рисуем (кроме
        базового изображения PictureBox)
        return;
    }
    // Рисуем первую линию (если есть первая
    точка и мы ждем вторую)
    if (drawnPoints.Count >= 1 && currentPhase ==
    DrawingPhase.WaitingForP2)
    {
        e.Graphics.DrawLine(drawingPen, drawnPoints[0],
        currentCursorPosition);
    }
    // Рисуем первую зафиксированную линию и вторую
    динамическую + дугу
    else if (drawnPoints.Count >= 2 && currentPhase ==
    DrawingPhase.WaitingForP3)
    {
        // Первая линия (P0-P1)
        e.Graphics.DrawLine(drawingPen, drawnPoints[0],
        drawnPoints[1]);
        // Вторая линия (P1-Cursor)
        e.Graphics.DrawLine(drawingPen, drawnPoints[1],
        currentCursorPosition);
        // Рисуем дугу и угол
        DrawAngleArc(e.Graphics, drawnPoints[0],
        drawnPoints[1], currentCursorPosition);
    }
    // Рисуем обе зафиксированные линии и дугу
    else if (drawnPoints.Count == 3 && (currentPhase ==
    DrawingPhase.DrawingCompleted || currentPhase ==
    DrawingPhase.WaitingForP1)) // WaitingForP1 - если пользователь снова
    нажал кнопку
    {
        // Первая линия (P0-P1)
        e.Graphics.DrawLine(drawingPen, drawnPoints[0],
        drawnPoints[1]);
        // Вторая линия (P1-P2)
        e.Graphics.DrawLine(drawingPen, drawnPoints[1],
        drawnPoints[2]);
        // Рисуем дугу и угол
        DrawAngleArc(e.Graphics, drawnPoints[0],
        drawnPoints[1], drawnPoints[2]);
    }
    // Если только первая точка поставлена, но фаза уже не
    WaitingForP2 (например, сразу нажали кнопку сброса)
    // или если просто очистили и еще ничего не начали
    рисовать, то ничего дополнительно не рисуем.
    }
    // Метод для расчета угла и обновления TextBox
    private void UpdateAngleDisplay(bool isFinal)
    {

```

```

        if (activeAngleTextBox == null) return;                                PointF
p0, pVertex, pEndArm;                if (drawnPoints.Count == 2 &&
currentPhase == DrawingPhase.WaitingForP3) // Динамический расчет для
второй линии
    {
        p0 = drawnPoints[0];
        pVertex = drawnPoints[1];
        pEndArm = currentCursorPoint;
    }
    else if (drawnPoints.Count == 3 && isFinal) // Финальный
расчет
    {
        p0 = drawnPoints[0];
        pVertex = drawnPoints[1];
        pEndArm = drawnPoints[2];
    }
    else
    {
        // activeAngleTextBox.Text = ""; // Очищаем, если
недостаточно точек
        return;
    }
    double angle =
CalculateAngleBetweenPoints(p0, pVertex, pEndArm);
    activeAngleTextBox.Text = angle.ToString("F3");
}
// Метод для рисования дуги угла
private void DrawAngleArc(Graphics g, PointF p0, PointF
pVertex, PointF pEndArm)
{
    // Проверка, что точки не совпадают с вершиной, чтобы
избежать ошибок Math.Atan2
    if (pVertex.Equals(p0) || pVertex.Equals(pEndArm))
    {
        return; // Невозможно построить угол
    }
    // Угол между вектором (pVertex -> p0) и осью
X
    double startAngleRad = Math.Atan2(p0.Y - pVertex.Y, p0.X -
pVertex.X);
    // Угол между вектором (pVertex -> pEndArm) и осью X
    double endAngleRad = Math.Atan2(pEndArm.Y - pVertex.Y,
pEndArm.X - pVertex.X);
    float startAngleDeg =
(float)(startAngleRad * 180.0 / Math.PI);
    float endAngleDeg = (float)(endAngleRad * 180.0 /
Math.PI);
    float sweepAngleDeg = endAngleDeg -
startAngleDeg;
    // Нормализация sweepAngle, чтобы он был в
диапазоне (-360, 360)
    // и соответствовал наименьшему углу между линиями
    if (sweepAngleDeg > 180f)
    {
        sweepAngleDeg -= 360f;
    }
    else if (sweepAngleDeg < -180f)
    {
        sweepAngleDeg += 360f;
    }
    // Если угол очень маленький, дуга может не
нарисоваться или выглядеть странно.

```

```

        // Также избегаем рисования дуги, если линии коллинеарны
        (угол 0 или 180).
        // Для CalculateAngleBetweenPoints это не проблема, но для
        DrawArc может быть.
        double calculatedAngle = CalculateAngleBetweenPoints(p0,
        pVertex, pEndArm);
        if (calculatedAngle > 0.1 && calculatedAngle < 179.9) //
        Не рисуем дугу для слишком маленьких/больших углов
        {
            RectangleF arcRect = new RectangleF(
                pVertex.X - ARC_RADIUS,
                pVertex.Y - ARC_RADIUS,
                2 * ARC_RADIUS,
                2 * ARC_RADIUS);
            g.DrawArc(drawingPen, arcRect, startAngleDeg,
            sweepAngleDeg);
        }
    } // Метод для вычисления угла между тремя точками
    (pVertex - вершина угла)
    public static double CalculateAngleBetweenPoints(PointF p0,
    PointF pVertex, PointF pEndArm)
    {
        // Векторы от вершины угла
        double v1x = p0.X - pVertex.X;
        double v1y = p0.Y - pVertex.Y;
        double v2x = pEndArm.X - pVertex.X;
        double v2y = pEndArm.Y - pVertex.Y; //
        Скалярное произведение
        double dotProduct = v1x * v2x + v1y * v2y; //
        Длины векторов
        double mag1 = Math.Sqrt(v1x * v1x + v1y * v1y);
        double mag2 = Math.Sqrt(v2x * v2x + v2y * v2y);
        // Если длина одного из векторов равна 0, угол не определен (или 0)
        if (mag1 == 0 || mag2 == 0)
            return 0.0; // Косинус угла
        double cosTheta = dotProduct / (mag1 * mag2);
        // Иногда из-за ошибок округления cosTheta может быть чуть > 1 или < -
        1
        cosTheta = Math.Max(-1.0, Math.Min(1.0, cosTheta));
        // Угол в радианах
        double angleRad = Math.Acos(cosTheta); // Угол
        в градусах
        return angleRad * (180.0 / Math.PI);
    }
    private void btn_next_3_Click(object sender,
    EventArgs e)
    {
        if (textBox_3_1_Angle_A.Text == "" ||
        textBox_3_2_Angle_B.Text == "")
        {
            MessageBox.Show("Требуемые значения не измерены!",
                "Ошибка",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
        else
        {

```

```

        SharedData.Angle_A =
Convert.ToDouble(textBox_3_1_Angle_A.Text);
        SharedData.Angle_B =
Convert.ToDouble(textBox_3_2_Angle_B.Text);
        Window_4 w4 = new Window_4();
        w4.Show();
        Hide();
    }
}
}
}Файл Window_9.cs
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.IO;
using System.Text;
using System.Windows.Forms; namespace Medic
{
    public partial class Window_9 : Form
    {
        private Image clueImage_10;
        private enum DrawingPhase
        {
            None, // Ничего не рисуется
            WaitingForP1, // Ожидание первой точки
            WaitingForP2, // Ожидание второй точки (первая линия
динамическая)
            WaitingForP3, // Ожидание третьей точки (вторая линия и
угол динамические)
            DrawingCompleted // Все три точки установлены
        }
        private DrawingPhase currentPhase =
DrawingPhase.None;
        private List<PointF> drawnPoints = new List<PointF>();
        private PointF currentCursorPoint; // Текущая позиция курсора
для динамического рисования
        private TextBox activeAngleTextBox = null; // TextBox для
вывода текущего угла
        private Pen drawingPen; // Перо для рисования
        private
const float ARC_RADIUS = 30f; // Радиус дуги для отображения угла
        public Window_9()
        {
            InitializeComponent();
            pictureBox9.Image = SharedData.Radiograph_4;
            LoadClueImage();
            btn_switch_9_1.Enabled = false;
            btn_switch_9_2.Enabled = false;
            label_9_4.Visible = false;
            DoubleBuffered = true;
            drawingPen = new Pen(Color.Yellow, 3);
        }
        private void LoadClueImage()
        {
            try
            {

```

```

        string clueImagePath =
Path.Combine(Application.StartupPath, "clue_10.png");
if (File.Exists(clueImagePath))
    {
        clueImage_10 = Image.FromFile(clueImagePath);
    }
else
    {
        MessageBox.Show("Файл clue_10.png не найден в
папке с программой!",
                        "Предупреждение",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Warning);
    }
}
catch (Exception ex)
{
    MessageBox.Show($"Ошибка при загрузке clue_10.png:
{ex.Message}",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
}
}
private void btn_select_4_Click(object sender,
EventArgs e)
{
    using (OpenFileDialog openFileDialog = new
OpenFileDialog())
    {
        openFileDialog.Title = "Выберите изображение";
        openFileDialog.Filter = "Файлы
изображений|*.png;*.jpg;*.jpeg|PNG файлы|*.png|JPEG
файлы|*.jpg;*.jpeg";
        if (openFileDialog.ShowDialog() ==
DialogResult.OK)
        {
            try
            {
                SharedData.Radiograph_4 =
(Bitmap) Image.FromFile(openFileDialog.FileName);
                pictureBox9.Image = SharedData.Radiograph_4;
                btn_switch_9_1.Enabled = true;
                btn_switch_9_2.Enabled = (clueImage_10 !=
null);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка при загрузке
изображения: {ex.Message}",
                                "Ошибка",
                                MessageBoxButtons.OK,
                                MessageBoxIcon.Error);
            }
        }
    }
}
private void btn_switch_9_1_Click(object sender,
EventArgs e)

```

```

        {
            pictureBox9.Image = SharedData.Radiograph_4;
        }
        private void btn_switch_9_2_Click(object sender,
EventArgs e)
        {
            pictureBox9.Image = clueImage_10;
            drawnPoints.Clear();
            pictureBox9.Invalidate();
        }
        private void btn_help_9_Click(object sender,
EventArgs e)
        {
            label_9_4.Visible = !label_9_4.Visible;
        }
        private void btn_measure_9_1_Click(object sender,
EventArgs e)
        {
            if (SharedData.Radiograph_4 != null)
            {
                if (pictureBox9.Image != SharedData.Radiograph_4)
                {
                    MessageBox.Show("Рентгенограммы 4 нет на экране!",
                        "Ошибка",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Error);
                }
                else StartMeasurement(textBox_9_1_Angle_E);
            }
            else
            {
                MessageBox.Show("Сначала загрузите рентгенограмму!",
                    "Ошибка",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Error);
            }
        }
        private void StartMeasurement(TextBox targetTextBox)
        {
            currentPhase = DrawingPhase.WaitingForP1;
            drawnPoints.Clear();
            activeAngleTextBox = targetTextBox;
            if (activeAngleTextBox != null)
            {
                activeAngleTextBox.Text = ""; // Очищаем поле угла
            }
            pictureBox9.Invalidate(); // Перерисовать PictureBox
(очистит предыдущие линии)
            // Убедимся, что PictureBox может получать фокус для
событий мыши, если нужно
            // pictureBox3.Focus(); // Обычно не требуется для
PictureBox
        } // Событие нажатия кнопки мыши на PictureBox
        private void pictureBox9_MouseDown(object sender,
MouseEventArgs e)
        {
            if (activeAngleTextBox == null || e.Button !=
MouseButtons.Left)
                return;
            switch (currentPhase)

```

```

        {
            case DrawingPhase.WaitingForP1:
                drawnPoints.Add(e.Location);
                currentPhase = DrawingPhase.WaitingForP2;
                currentCursorPoint = e.Location; // Инициализируем
для первого кадра
                break;
            case DrawingPhase.WaitingForP2:
                drawnPoints.Add(e.Location);
                currentPhase = DrawingPhase.WaitingForP3;
                currentCursorPoint = e.Location; // Инициализируем
для первого кадра
                break;
            case DrawingPhase.WaitingForP3:
                drawnPoints.Add(e.Location);
                currentPhase = DrawingPhase.DrawingCompleted;
                UpdateAngleDisplay(true); // Обновить угол в
TextBox окончательно
                break;
        }
        pictureBox9.Invalidate(); // Запросить перерисовку
    } // Событие перемещения мыши над PictureBox
    private void pictureBox9_MouseMove(object sender,
MouseEventArgs e)
    {
        if (activeAngleTextBox == null)
            return;
        currentCursorPoint = e.Location;
        if (currentPhase == DrawingPhase.WaitingForP2 || currentPhase ==
DrawingPhase.WaitingForP3)
        {
            if (currentPhase == DrawingPhase.WaitingForP3 &&
drawnPoints.Count == 2)
            {
                UpdateAngleDisplay(false); // Динамическое
обновление угла в TextBox
            }
            pictureBox9.Invalidate(); // Запросить перерисовку для
динамического отображения
        }
    } // Событие перерисовки PictureBox
    private void pictureBox9_Paint(object sender, PaintEventArgs
e)
    {
        // Устанавливаем высокое качество рендеринга
        e.Graphics.SmoothingMode = SmoothingMode.AntiAlias;
        if (activeAngleTextBox == null && currentPhase == DrawingPhase.None)
        {
            // Если не в режиме измерения, ничего не рисуем (кроме
базового изображения PictureBox)
            return;
        } // Рисуем первую линию (если есть первая
точка и мы ждем вторую)
        if (drawnPoints.Count >= 1 && currentPhase ==
DrawingPhase.WaitingForP2)
        {

```

```

        e.Graphics.DrawLine(drawingPen, drawnPoints[0],
currentCursorPoint);
    }
    // Рисуем первую зафиксированную линию и вторую
динамическую + дугу
    else if (drawnPoints.Count >= 2 && currentPhase ==
DrawingPhase.WaitingForP3)
    {
        // Первая линия (P0-P1)
        e.Graphics.DrawLine(drawingPen, drawnPoints[0],
drawnPoints[1]);
        // Вторая линия (P1-Cursor)
        e.Graphics.DrawLine(drawingPen, drawnPoints[1],
currentCursorPoint);
        // Рисуем дугу и угол
        DrawAngleArc(e.Graphics, drawnPoints[0],
drawnPoints[1], currentCursorPoint);
    }
    // Рисуем обе зафиксированные линии и дугу
    else if (drawnPoints.Count == 3 && (currentPhase ==
DrawingPhase.DrawingCompleted || currentPhase ==
DrawingPhase.WaitingForP1)) // WaitingForP1 - если пользователь снова
нажал кнопку
    {
        // Первая линия (P0-P1)
        e.Graphics.DrawLine(drawingPen, drawnPoints[0],
drawnPoints[1]);
        // Вторая линия (P1-P2)
        e.Graphics.DrawLine(drawingPen, drawnPoints[1],
drawnPoints[2]);
        // Рисуем дугу и угол
        DrawAngleArc(e.Graphics, drawnPoints[0],
drawnPoints[1], drawnPoints[2]);
    }
    // Если только первая точка поставлена, но фаза уже не
WaitingForP2 (например, сразу нажали кнопку сброса)
    // или если просто очистили и еще ничего не начали
рисовать, то ничего дополнительно не рисуем.
    } // Метод для расчета угла и обновления TextBox
    private void UpdateAngleDisplay(bool isFinal)
    {
        if (activeAngleTextBox == null) return;
        p0, pVertex, pEndArm;
        if (drawnPoints.Count == 2 &&
currentPhase == DrawingPhase.WaitingForP3) // Динамический расчет для
второй линии
        {
            p0 = drawnPoints[0];
            pVertex = drawnPoints[1];
            pEndArm = currentCursorPoint;
        }
        else if (drawnPoints.Count == 3 && isFinal) // Финальный
расчет
        {
            p0 = drawnPoints[0];
            pVertex = drawnPoints[1];
            pEndArm = drawnPoints[2];
        }
    }
}

```

```

    }
    else
    {
        // activeAngleTextBox.Text = ""; // Очищаем, если
недостаточно точек
        return;
    }
    double angle =
CalculateAngleBetweenPoints(p0, pVertex, pEndArm);
    activeAngleTextBox.Text = angle.ToString("F3");
}
// Метод для рисования дуги угла
private void DrawAngleArc(Graphics g, PointF p0, PointF
pVertex, PointF pEndArm)
{
    // Проверка, что точки не совпадают с вершиной, чтобы
избежать ошибок Math.Atan2
    if (pVertex.Equals(p0) || pVertex.Equals(pEndArm))
    {
        return; // Невозможно построить угол
    }
    // Угол между вектором (pVertex -> p0) и осью
X
    double startAngleRad = Math.Atan2(p0.Y - pVertex.Y, p0.X -
pVertex.X);
    // Угол между вектором (pVertex -> pEndArm) и осью X
    double endAngleRad = Math.Atan2(pEndArm.Y - pVertex.Y,
pEndArm.X - pVertex.X);
    float startAngleDeg =
(float)(startAngleRad * 180.0 / Math.PI);
    float endAngleDeg = (float)(endAngleRad * 180.0 /
Math.PI);
    float sweepAngleDeg = endAngleDeg -
startAngleDeg;
    // Нормализация sweepAngle, чтобы он был в
диапазоне (-360, 360)
    // и соответствовал наименьшему углу между линиями
    if (sweepAngleDeg > 180f)
    {
        sweepAngleDeg -= 360f;
    }
    else if (sweepAngleDeg < -180f)
    {
        sweepAngleDeg += 360f;
    }
    // Если угол очень маленький, дуга может не
нарисоваться или выглядеть странно.
    // Также избегаем рисования дуги, если линии коллинеарны
(угол 0 или 180).
    // Для CalculateAngleBetweenPoints это не проблема, но для
DrawArc может быть.
    double calculatedAngle = CalculateAngleBetweenPoints(p0,
pVertex, pEndArm);
    if (calculatedAngle > 0.1 && calculatedAngle < 179.9) //
Не рисуем дугу для слишком маленьких/больших углов
    {
        RectangleF arcRect = new RectangleF(
            pVertex.X - ARC_RADIUS,
            pVertex.Y - ARC_RADIUS,
            2 * ARC_RADIUS,
            2 * ARC_RADIUS);
        g.DrawArc(drawingPen, arcRect, startAngleDeg,
sweepAngleDeg);
    }
}

```

```

    }
} // Метод для вычисления угла между тремя точками
(pVertex - вершина угла)
public static double CalculateAngleBetweenPoints(PointF p0,
PointF pVertex, PointF pEndArm)
{
    // Векторы от вершины угла
    double v1x = p0.X - pVertex.X;
    double v1y = p0.Y - pVertex.Y;
    double v2x = pEndArm.X - pVertex.X;
    double v2y = pEndArm.Y - pVertex.Y; //
    Скалярное произведение
    double dotProduct = v1x * v2x + v1y * v2y; //
    Длины векторов
    double mag1 = Math.Sqrt(v1x * v1x + v1y * v1y);
    double mag2 = Math.Sqrt(v2x * v2x + v2y * v2y);
    // Если длина одного из векторов равна 0, угол не определен (или 0)
    if (mag1 == 0 || mag2 == 0)
        return 0.0; // Косинус угла
    double cosTheta = dotProduct / (mag1 * mag2);
    // Иногда из-за ошибок округления cosTheta может быть чуть > 1 или < -
    1
    cosTheta = Math.Max(-1.0, Math.Min(1.0, cosTheta));
    // Угол в радианах
    double angleRad = Math.Acos(cosTheta); // Угол
    в градусах
    return angleRad * (180.0 / Math.PI);
}
public static string FormatFloat(double value)
{
    return value.ToString("F2");
}
public static string
GenerateVerdictFor13AndOlder(string PHIO, int year, double ICAS,
double angle_A, double angle_B, double angle_V, double angle_Vibert,
double angle_D, double angle_E, double dist_D, double dist_W, double
UOB, double Dh, double AK, double SLN, double SPN, double HP, double
BP, double trans)
{
    StringBuilder sb = new StringBuilder();
    sb.Append("{\\rtf1\\ansi\\ansicpg1251\\deff0\\n");
    sb.Append("{\\fonttbl{\\f0 Times New Roman;}}\\n");
    sb.Append("\\pard\\sl360\\slmult1\\n");
    sb.Append("\\f0\\fs28\\n"); if (ICAS >= 1.0 &&
ICAS <= 1.7)
    {
        if (angle_A <= 35)
        {
            if (angle_B >= 15)
            {
                sb.Append("\\qc \\fs32 Вердикт 1\\par \\par");
                sb.Append("\\ql \\fs28 " + PHIO +
"\\par");
                sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par \\par");
                sb.Append("\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par");

```

```

        sb.Append("\\ql \\fs28 Ацетабулярный
коэффициент АК равен " + FormatFloat(AK) + "\\par Показана остеотомия
таза.\\par");

        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента равна " + FormatFloat(SLN) +
"\\par");

        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента равна " + FormatFloat(SPN) +
"\\par");

        if (angle_V >= 10 && angle_V <= 20)
        {
        }
        else if (angle_V > 20)
        {
            sb.Append("\\ql \\fs28 А Наружная ротация
ацетабулярного фрагмента НР равна " + FormatFloat(HP) + "\\par");
        }
        else if (angle_V < 10)
        {
            sb.Append("\\ql \\fs28 Б Внутренняя
ротация ацетабулярного фрагмента ВР равна " + FormatFloat(BP) +
"\\par");
        }
    }
    else
    {
        sb.Append("\\qc \\fs32 Вердикт 0\\par \\par");
        sb.Append("\\ql \\fs28      " + PHIO +
"\\par");

        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par \\par");
        sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
    }
}
else
{
    sb.Append("\\qc \\fs32 Вердикт 0\\par \\par");
    sb.Append("\\ql \\fs28      " + PHIO + "\\par");
    sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par \\par");
    sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
}
}
else if (ICAS > 1.7)
{
    string text = "\\ql \\fs28 Ацетабулярный коэффициент
АК равен " + FormatFloat(AK) + "\\par";
    if (angle_A > 35)
    {
        text += "\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par";
        if (angle_V >= 10 && angle_V <= 20)

```

```

        {
            sb.Append("\\qc \\fs32 Вердикт 2\\par\\par");
            sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
            sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
            sb.Append(text);
            sb.Append("\\ql \\fs28 Показана неполная
периацетабулярная остеотомия (ПАО)\\par");
            sb.Append("\\ql \\fs28 Наклон ацетабулярного
фрагмента во фронтальной и сагиттальной плоскости: \\par");
            sb.Append("\\ql \\fs28 Степень латерального
наклона (фронтальная плоскость) равна " + FormatFloat(angle_B) +
"\\par");
            sb.Append("\\ql \\fs28 Степень латерального
наклона соответствует углу наклона опорной поверхности Б\\par");
            sb.Append("\\ql \\fs28 Степень переднего
наклона (сагиттальная плоскость) равна " + FormatFloat(SPN) +
"\\par");
        }
        else
        {
            sb.Append("\\qc \\fs32 Вердикт 3\\par\\par");
            sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
            sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
            sb.Append(text);
            sb.Append("Показана остеотомия таза + неполная
периацетабулярная остеотомия (ПАО)\\par");
            sb.Append("За счет остеотомии таза
осуществляется наклон в сагиттальной и горизонтальной
плоскости\\par");
            sb.Append("Сагиттальная плоскость (передний
наклон) Степень переднего наклона фрагмента равна " + FormatFloat(SPN)
+ "\\par");
            if (angle_V > 20)
            {
                sb.Append("\\ql \\fs28 А Горизонтальная
плоскость. \\par");
                sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента НР равна " + FormatFloat(НР) + "\\par");
            }
            else if (angle_V < 10)
            {
                sb.Append("\\ql \\fs28 Б Горизонтальная
плоскость.\\par");
                sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента ВР равна " + FormatFloat(ВР) + "\\par");
            }
            sb.Append("\\ql \\fs28 За счет неполной ПАО
осуществляется латеральный наклон\\par");
            sb.Append("\\ql \\fs28 Степень латерального
наклона соответствует углу наклона опорной поверхности Б = " +
FormatFloat(angle_B) + "\\par"); //
        }
    }

```

```

    }
    else if (angle_A <= 35)
    {
        sb.Append("\\qc \\fs32 Вердикт 4\\par\\par");
        sb.Append("\\ql \\fs28      " + PHI0 + "\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append(text);
        sb.Append("\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par");
        sb.Append("\\ql \\fs28 Показана остеотомия таза +
неполная периацетабулярная остеотомия\\par");
        if (angle_V >= 10 && angle_V <= 20)
        {
            sb.Append("\\ql \\fs28 А. За счет остеотомии
таза осуществляется наклон во фронтальной и сагиттальной
плоскости\\par");
            sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон): \\par");
            sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента (СЛН) равна " + FormatFloat(SLN) +
"\\par");
            sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон): \\par");
            sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
        }
        else if (angle_V > 20)
        {
            sb.Append("\\ql \\fs28 Б. За счет остеотомии
таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
            sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
            sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента (СЛН) равна " + FormatFloat(SLN) +
"\\par");
            sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
            sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
            sb.Append("\\ql \\fs28 В горизонтальной
плоскости:\\par");
            sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента НР равна " + FormatFloat(HR) + "\\par");
        }
        else if (angle_V < 10)
        {
            sb.Append("\\ql \\fs28 В. За счет остеотомии
таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
            sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");

```

```

        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента равна " + FormatFloat(SLN) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
        sb.Append("\\ql \\fs28 В горизонтальной
плоскости:\\par");
        sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента ВР равна " + FormatFloat(BP) + "\\par");
    }
    sb.Append("\\ql \\fs28 За счет неполной
периацетабулярной остеотомии (ПАО) осуществляется дополнительный
латеральный наклон\\par");
    sb.Append("\\ql \\fs28 Степень дополнительного
латерального наклона = " + FormatFloat((angle_B - angle_E)) +
"\\par");
    }
}
else if (ICAS >= 0.8 && ICAS <= 0.99)
{
    if (angle_A > 35)
    {
        if (angle_B >= 15 && angle_B <= 20)
        {
            sb.Append("\\qc \\fs32 Вердикт 5\\par\\par");
            sb.Append("\\ql \\fs28      " + РНЮ +
"\\par");
            sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
            sb.Append("\\ql \\fs28 Г. Ацетабулярный
коэффициент равен " + FormatFloat(AK) + "\\par");
            sb.Append("\\ql \\fs28 Показана
ацетабулопластика\\par");
            sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
            sb.Append("\\ql \\fs28 Величина трансплантата:
" + FormatFloat(trans) + "\\par");
        }
        else
        {
            sb.Append("\\qc \\fs32 Вердикт 0\\par\\par");
            sb.Append("\\ql \\fs28      " + РНЮ +
"\\par");
            sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
            sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
        }
    }
}
else if (angle_A <= 35)
{

```

```

        if (angle_B >= 15)
        {
            sb.Append("\\qc \\fs32 Вердикт 5\\par\\par");
            sb.Append("\\ql \\fs28      " + PHI0 +
"\\par");
            sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
            sb.Append("\\ql \\fs28 Ацетабулярный
коэффициент равен " + FormatFloat(AK) + "\\par");
            sb.Append("\\ql \\fs28 Показана остеотомия
таза + ацетабулопластика\\par");
            if (angle_V >= 10 && angle_V <= 20)
            {
                sb.Append("\\ql \\fs28 А. За счет
osteotomii таза осуществляется наклон во фронтальной и сагиттальной
плоскости\\par");
                sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
                sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента соответствует величине
угла наклона опорной поверхности Б = " + FormatFloat(angle_B) +
"\\par");
                sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
                sb.Append("\\ql \\fs28 Степень переднего
наклона фрагмента равна величине угла Е = " + FormatFloat(angle_E) +
"\\par");
                sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
                sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
            }
            else if (angle_V > 20)
            {
                sb.Append("\\ql \\fs28 Б. За счет
osteotomii таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
                sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
                sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента соответствует величине
угла наклона опорной поверхности Б = " + FormatFloat(angle_B) +
"\\par");
                sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
                sb.Append("\\ql \\fs28 Степень переднего
наклона фрагмента равна величине угла Е = " + FormatFloat(angle_E) +
"\\par");
                sb.Append("\\ql \\fs28 Горизонтальная
плоскость:\\par");
                sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента: " + FormatFloat(HP) + "\\par");
                sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
            }
        }
    }
}

```

```

        sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
    }
    else if (angle_V < 10)
    {
        sb.Append("\\ql \\fs28 В. За счет
osteotомии таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
        sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
        sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента соответствует величине
угла наклона опорной поверхности Б = " + FormatFloat(angle_B) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона фрагмента равна величине угла Е = " + FormatFloat(angle_E) +
"\\par");
        sb.Append("\\ql \\fs28 Горизонтальная
плоскость:\\par");
        sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента: " + FormatFloat(BP) + "\\par");
        sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
        sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
    }
}
else
{
    sb.Append("\\qc \\fs32 Вердикт 0\\par\\par");
    sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
    sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
    sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
}
}
}
    sb.Append("\\par");
    sb.Append("}");
    return sb.ToString();
}
    public static string GenerateVerdictFor10To13(string
PHIO, int year, double ICAS, double angle_A, double angle_B, double
angle_V, double angle_Vibert, double angle_D, double angle_E, double
dist_D, double dist_W, double UOB, double Dh, double AK, double SLN,
double SPN, double HP, double BP, double trans)
{
    double FPN = angle_B - 10;
    StringBuilder sb = new StringBuilder();
    sb.Append("{\\rtf1\\ansi\\ansicpg1251\\deff0\\n");
    sb.Append("{\\fonttbl{\\f0 Times New Roman;}}\\n");
    sb.Append("\\pard\\sl360\\slmult1\\n");

```

```

sb.Append("\\f0\\fs28\\n");
ICAS <= 1.7)
{
    if (angle_A <= 35)
    {
        if (angle_B >= 15)
        {
            sb.Append("\\qc \\fs32 Вердикт 1\\par \\par");
            sb.Append("\\ql \\fs28 " + PHIO +
"\\par");
            sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par \\par");
            sb.Append("\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par");
            sb.Append("\\ql \\fs28 Ацетабулярный
коэффициент АК равен " + FormatFloat(AK) + "\\par Показана остеотомия
таза (лонно-подвздошная остеотомия при сохранении ЛСС). \\par");
            sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента равна " + FormatFloat(SLN) +
"\\par");
            sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента равна " + FormatFloat(SPN) +
"\\par");
            if (angle_V >= 10 && angle_V <= 20)
            {
            }
            else if (angle_V > 20)
            {
                sb.Append("\\ql \\fs28 А Наружная ротация
ацетабулярного фрагмента НР равна " + FormatFloat(HR) + "\\par");
            }
            else if (angle_V < 10)
            {
                sb.Append("\\ql \\fs28 Б Внутренняя
ротация ацетабулярного фрагмента ВР равна " + FormatFloat(BR) +
"\\par");
            }
        }
        else
        {
            sb.Append("\\qc \\fs32 Вердикт 0\\par \\par");
            sb.Append("\\ql \\fs28 " + PHIO +
"\\par");
            sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par \\par");
            sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
        }
    }
    else
    {
        sb.Append("\\qc \\fs32 Вердикт 0\\par \\par");
        sb.Append("\\ql \\fs28 " + PHIO + "\\par");
        sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par \\par");
    }
}

```

```

        sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
    }
}
else if (ICAS > 1.7)
{
    string text = "\\ql \\fs28 Ацетабулярный коэффициент
АК равен " + FormatFloat(AK) + "\\par";
    if (angle_A > 35)
    {
        text += "\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par";
        if (angle_V >= 10 && angle_V <= 20)
        {
            sb.Append("\\qc \\fs32 Вердикт 2\\par\\par");
            sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
            sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
            sb.Append(text);
            sb.Append("\\ql \\fs28 Показана неполная
периацетабулярная остеотомия (ПАО)\\par");
            sb.Append("\\ql \\fs28 Наклон ацетабулярного
фрагмента во фронтальной и сагиттальной плоскости: \\par");
            sb.Append("\\ql \\fs28 Степень латерального
наклона (фронтальная плоскость) равна " + FormatFloat(FPN) + "\\par");
            sb.Append("\\ql \\fs28 Степень переднего
наклона (сагиттальная плоскость) равна " + FormatFloat(SPN) +
"\\par");
        }
    }
    else
    {
        sb.Append("\\qc \\fs32 Вердикт 3\\par\\par");
        sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append(text);
        sb.Append("Показана остеотомия таза (лонно-
подвздошная остеотомия при сохранении ЛСС) + неполная
периацетабулярная остеотомия (ПАО)\\par");
        sb.Append("За счет остеотомии таза
осуществляется наклон в сагиттальной и горизонтальной
плоскости\\par");
        sb.Append("Сагиттальная плоскость (передний
наклон) Степень переднего наклона фрагмента равна " + FormatFloat(SPN)
+ "\\par");
        if (angle_V > 20)
        {
            sb.Append("\\ql \\fs28 А Горизонтальная
плоскость. \\par");
            sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента НР равна " + FormatFloat(HR) + "\\par");
        }
    }
    else if (angle_V < 10)

```

```

        {
            sb.Append("\\ql \\fs28 В Горизонтальная
плоскость.\\par");
            sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента ВР равна " + FormatFloat(BP) + "\\par");
        }
        sb.Append("\\ql \\fs28 За счет неполной ПАО
осуществляется латеральный наклон\\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона соответствует углу наклона опорной поверхности В = " +
FormatFloat(FPN) + "\\par"); //
    }
}
else if (angle_A <= 35)
{
    sb.Append("\\qc \\fs32 Вердикт 4\\par\\par");
    sb.Append("\\ql \\fs28 " + РНЮ + "\\par");
    sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par\\par");
    sb.Append(text);
    sb.Append("\\ql \\fs28 Угол наклона опорной
поверхности (В) равен " + FormatFloat(angle_B) + "\\par");
    sb.Append("\\ql \\fs28 Показана остеотомия таза
(лонно-подвздошная остеотомия при сохранении ЛСС) + неполная
периацетабулярная остеотомия\\par");
    if (angle_V >= 10 && angle_V <= 20)
    {
        sb.Append("\\ql \\fs28 А. За счет остеотомии
таза осуществляется наклон во фронтальной и сагиттальной
плоскости\\par");
        sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон): \\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента (СЛН) равна " + FormatFloat(SLN) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон): \\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
    }
    else if (angle_V > 20)
    {
        sb.Append("\\ql \\fs28 Б. За счет остеотомии
таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
        sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента (СЛН) равна " + FormatFloat(SLN) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
    }
}

```

```

        sb.Append("\\ql \\fs28 В горизонтальной
плоскости:\\par");
        sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента НР равна " + FormatFloat(НР) + "\\par");
    }
    else if (angle_V < 10)
    {
        sb.Append("\\ql \\fs28 В. За счет остеотомии
таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
        sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента равна " + FormatFloat(SLN) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
        sb.Append("\\ql \\fs28 В горизонтальной
плоскости:\\par");
        sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента ВР равна " + FormatFloat(ВР) + "\\par");
    }
    sb.Append("\\ql \\fs28 За счет неполной
периацетабулярной остеотомии (ПАО) осуществляется дополнительный
латеральный наклон\\par");
    sb.Append("\\ql \\fs28 Степень дополнительного
латерального наклона = " + FormatFloat((angle_B - angle_E)) +
"\\par");
    }
}
else if (ICAS >= 0.8 && ICAS <= 0.99)
{
    if (angle_A > 35)
    {
        if (angle_B >= 15 && angle_B <= 20)
        {
            sb.Append("\\qc \\fs32 Вердикт 5\\par\\par");
            sb.Append("\\ql \\fs28      " + РНЮ +
"\\par");
            sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
            sb.Append("\\ql \\fs28 Г. Ацетабулярный
коэффициент равен " + FormatFloat(АК) + "\\par");
            sb.Append("\\ql \\fs28 Показана
ацетабулопластика\\par");
            sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
            sb.Append("\\ql \\fs28 Величина трансплантата:
" + FormatFloat(trans) + "\\par");
        }
        else
        {

```

```

        sb.Append("\\qc \\fs32 Вердикт 0\\par\\par");
        sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
    }
}
else if (angle_A <= 35)
{
    if (angle_B >= 15)
    {
        sb.Append("\\qc \\fs32 Вердикт 5\\par\\par");
        sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append("\\ql \\fs28 Ацетабулярный
коэффициент равен " + FormatFloat(AK) + "\\par");
        sb.Append("\\ql \\fs28 Показана остеотомия
таза (лонно-подвздошная остеотомия при сохранении ЛСС) +
ацетабулопластика\\par");
        if (angle_V >= 10 && angle_V <= 20)
        {
            sb.Append("\\ql \\fs28 А. За счет
osteotomy таза осуществляется наклон во фронтальной и сагиттальной
плоскости\\par");
            sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
            sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента = " + FormatFloat(FPN) +
"\\par");
            sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
            sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента равна величине угла Е = " +
FormatFloat(angle_E) + "\\par");
            sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
            sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
        }
        else if (angle_V > 20)
        {
            sb.Append("\\ql \\fs28 Б. За счет
osteotomy таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
            sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
            sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента = " + FormatFloat(FPN) +
"\\par");

```

```

        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента равна величине угла E = " +
FormatFloat(angle_E) + "\\par");
        sb.Append("\\ql \\fs28 Горизонтальная
плоскость:\\par");
        sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента: " + FormatFloat(HP) + "\\par");
        sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
        sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
    }
    else if (angle_V < 10)
    {
        sb.Append("\\ql \\fs28 В. За счет
остеотомии таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
        sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
        sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента = " + FormatFloat(FPN) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента равна величине угла E = " +
FormatFloat(angle_E) + "\\par");
        sb.Append("\\ql \\fs28 Горизонтальная
плоскость:\\par");
        sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента: " + FormatFloat(BP) + "\\par");
        sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
        sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
    }
    else
    {
        sb.Append("\\qc \\fs32 Вердикт 0\\par\\par");
        sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности B < 15°\\par В данном случае реконструктивная операция не
показана\\par");
    }
    }
    sb.Append("\\par");
    sb.Append("}");
    return sb.ToString();

```

```

    }        public static string GenerateVerdictFor7To9(string
PHIO, int year, double ICAS, double angle_A, double angle_B, double
angle_V, double angle_Vibert, double angle_D, double angle_E, double
dist_D, double dist_W, double UOB, double Dh, double AK, double SLN,
double SPN, double HP, double BP, double trans)
    {
        double FPN = angle_B - 10;
        StringBuilder sb = new StringBuilder();
        sb.Append("{\\rtf1\\ansi\\ansicpg1251\\deff0\\n");
        sb.Append("{\\fonttbl{\\f0 Times New Roman;}}\\n");
        sb.Append("\\pard\\sl360\\slmult1\\n");
        sb.Append("\\f0\\fs28\\n");          if (ICAS >= 1.0 &&
ICAS <= 1.7)
        {
            if (angle_A <= 35 && angle_A >= 32)
            {
                if (angle_B >= 25 && angle_B <= 35)
                {
                    sb.Append("\\qc \\fs32 Вердикт 1\\par \\par");
                    sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
                    sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par \\par");
                    sb.Append("\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par");
                    sb.Append("\\ql \\fs28 Ацетабулярный
коэффициент АК равен " + FormatFloat(AK) + "\\par");
                    sb.Append("\\ql \\fs28 Показана Г-образная
остеотомия подвздошной кости. \\par");
                    sb.Append("\\ql \\fs28 Наклон впадины во
фронтальной (латеральный наклон) и сагиттальной (передний наклон)
плоскости \\par");
                    sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента равна " + FormatFloat(SLN) +
"\\par");
                    sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
                }
                else
                {
                    sb.Append("\\qc \\fs32 Вердикт 0\\par \\par");
                    sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
                    sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par \\par");
                    sb.Append("\\qc \\fs32 Измеренные параметры
выходят за пределы допустимого диапазона \\par В данном случае
реконструктивная операция не показана\\par");
                }
            }
            else if (angle_B > 35)
            {
                sb.Append("\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par");
            }
        }
    }

```

```

        sb.Append("\\ql \\fs28 Ацетабулярный коэффициент
АК равен " + FormatFloat(AK) + "\\par");
        sb.Append("\\ql \\fs28 Показана остеотомия таза
(лонно-подвздошная остеотомия при сохранении ЛСС). \\par");
        sb.Append("\\ql \\fs28 Наклон впадины во
фронтальной (латеральный наклон), сагиттальной (передний наклон) и
горизонтальной плоскости \\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента равна " + FormatFloat(SLN) +
"\\par");
        sb.Append("\\ql \\fs28 Степень переднего наклона
ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) + "\\par");
    }
    else
    {
        sb.Append("\\qc \\fs32 Вердикт 0\\par \\par");
        sb.Append("\\ql \\fs28 " + РНЮ + "\\par");
        sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par \\par");
        sb.Append("\\qc \\fs32 Угол наклона опорной
поверхности Б < 15°\\par В данном случае реконструктивная операция не
показана\\par");
    }
}
else if (ICAS > 1.7)
{
    string text = "\\ql \\fs28 Ацетабулярный коэффициент
АК равен " + FormatFloat(AK) + "\\par";
    text += "\\ql \\fs28 Угол наклона опорной поверхности
(Б) равен " + FormatFloat(angle_B) + "\\par";
    if (angle_A > 35)
    {
        sb.Append("\\qc \\fs32 Вердикт 2\\par\\par");
        sb.Append("\\ql \\fs28 " + РНЮ + "\\par");
        sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append(text);
        sb.Append("\\ql \\fs28 Показана неполная
периацетабулярная остеотомия (ПАО)\\par");
        sb.Append("\\ql \\fs28 Наклон ацетабулярного
фрагмента во фронтальной и сагиттальной плоскости: \\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона (фронтальная плоскость) равна " + FormatFloat(FPN) + "\\par");
        sb.Append("\\ql \\fs28 Степень переднего наклона
(сагиттальная плоскость) равна " + FormatFloat(SPN) + "\\par");
    }
    else if (angle_A <= 35 && angle_A >= 30)
    {
        sb.Append("\\qc \\fs32 Вердикт 3\\par\\par");
        sb.Append("\\ql \\fs28 " + РНЮ + "\\par");
        sb.Append("\\ql \\fs28 Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append(text);
        sb.Append("\\ql \\fs28 Показана поперечная
osteotomy подвздошной кости + неполная периацетабулярная остеотомия
(ПАО) \\par");
    }
}

```

```

sb.Append("\\ql \\fs28 За счет поперечной
osteотомии подвздошной кости осуществляется наклон во фронтальной и
сагиттальной плоскости \\par");
sb.Append("\\ql \\fs28 Во фронтальной плоскости
(латеральный наклон) \\par");
sb.Append("\\ql \\fs28 За счет поперечной
osteотомии подвздошной кости осуществляется наклон во фронтальной и
сагиттальной плоскости \\par");
sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента = " + FormatFloat(SLN) + "\\par");
sb.Append("\\ql \\fs28 В сагиттальной плоскости
(передний наклон) \\par");
sb.Append("\\ql \\fs28 Степень переднего наклона
фрагмента равна " + FormatFloat(SPN) + "\\par");
sb.Append("\\ql \\fs28 За счет неполной ПАО
осуществляется дополнительный латеральный наклон \\par");
sb.Append("\\ql \\fs28 Степень дополнительного
латерального наклона:" + FormatFloat((angle_B - angle_E)) + "\\par");
}
else if (angle_A < 30)
{
text += "\\ql \\fs28 Показана остеотомия таза
(лонно-подвздошная остеотомия при сохранении ЛСС) + неполная
периацетабулярная остеотомия \\par";
if (angle_V >= 10 && angle_V <= 20)
{
sb.Append("\\ql \\fs28 А. За счет остеотомии
таза осуществляется наклон во фронтальной и сагиттальной
плоскости\\par");
sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон): \\par");
sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента (СЛН) равна " + FormatFloat(SLN) +
"\\par");
sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон): \\par");
sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
}
else if (angle_V > 20)
{
sb.Append("\\ql \\fs28 Б. За счет остеотомии
таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента (СЛН) равна " + FormatFloat(SLN) +
"\\par");
sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
}
}

```

```

        sb.Append("\\ql \\fs28 В горизонтальной
плоскости:\\par");
        sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента НР равна " + FormatFloat(НР) + "\\par");
    }
    else if (angle_V < 10)
    {
        sb.Append("\\ql \\fs28 В. За счет остеотомии
таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
        sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента равна " + FormatFloat(SLN) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента (СПН) равна " + FormatFloat(SPN) +
"\\par");
        sb.Append("\\ql \\fs28 В горизонтальной
плоскости:\\par");
        sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента ВР равна " + FormatFloat(ВР) + "\\par");
    }
    sb.Append("\\ql \\fs28 За счет неполной
периацетабулярной остеотомии (ПАО) осуществляется дополнительный
латеральный наклон\\par");
    sb.Append("\\ql \\fs28 Степень дополнительного
латерального наклона = " + FormatFloat((angle_B - angle_E)) +
"\\par");
    }
}
else if (ICAS >= 0.8 && ICAS <= 0.99)
{
    if (angle_A > 35)
    {
        sb.Append("\\qc \\fs32 Вердикт 5\\par\\par");
        sb.Append("\\ql \\fs28      " + РНЮ + "\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append("\\ql \\fs28 Г. Ацетабулярный
коэффициент равен " + FormatFloat(АК) + "\\par");
        sb.Append("\\ql \\fs28 Показана
ацетабулопластика\\par");
        sb.Append("\\ql \\fs28 За счет ацетабулопластики
осуществляется дополнительное латеральное покрытие наклона\\par");
        sb.Append("\\ql \\fs28 Величина трансплантата: " +
FormatFloat(trans) + "\\par");
    }
    else if (angle_A < 32)
    {
        if (angle_B > 35)
        {
            sb.Append("\\qc \\fs32 Вердикт 5\\par\\par");

```

```

sb.Append("\\ql \\fs28      " + РНЮ +
"\\par");
sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
sb.Append("\\ql \\fs28 Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par");
sb.Append("\\ql \\fs28 Ацетабулярный
коэффициент равен " + FormatFloat(AK) + "\\par");
sb.Append("\\ql \\fs28 Показана остеотомия
таза (лонно-подвздошная остеотомия при сохранении ЛСС) +
ацетабулопластика \\par");
if (angle_V >= 10 && angle_V <= 20)
{
sb.Append("\\ql \\fs28 А. За счет
osteotomii таза осуществляется наклон во фронтальной и сагиттальной
плоскости\\par");
sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента = " + FormatFloat(FPN) +
"\\par");
sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон)\\par");
sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента = " + FormatFloat(angle_E) +
"\\par");
sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
}
else if (angle_V > 20)
{
sb.Append("\\ql \\fs28 Б. За счет
osteotomii таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента = " + FormatFloat(FPN) +
"\\par");
sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон)\\par");
sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента = " + FormatFloat(angle_E) +
"\\par");
sb.Append("\\ql \\fs28 Горизонтальная
плоскость:\\par");
sb.Append("\\ql \\fs28 Наружная ротация
ацетабулярного фрагмента: " + FormatFloat(HP) + "\\par");
sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
}

```

```

    }
    else if (angle_V < 10)
    {
        sb.Append("\\ql \\fs28 В. За счет
osteотомии таза осуществляется наклон во фронтальной, сагиттальной и
горизонтальной плоскости\\par");
        sb.Append("\\ql \\fs28 Во фронтальной
плоскости (латеральный наклон):\\par");
        sb.Append("\\ql \\fs28 Степень
латерального наклона ацетабулярного фрагмента = " + FormatFloat(FPN) +
"\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон):\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента = " + FormatFloat(angle_E) +
"\\par");
        sb.Append("\\ql \\fs28 Горизонтальная
плоскость:\\par");
        sb.Append("\\ql \\fs28 Внутренняя ротация
ацетабулярного фрагмента: " + FormatFloat(BP) + "\\par");
        sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
наклона\\par");
        sb.Append("\\ql \\fs28 Величина
трансплантата: " + FormatFloat(trans) + "\\par");
    }
}
else
{
    sb.Append("\\qc \\fs32 Вердикт 0\\par\\par");
    sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
    sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
    sb.Append("\\qc \\fs32 Измеренные параметры
выходят за пределы допустимого диапазона \\par В данном случае
реконструктивная операция не показана\\par");
}
}
else if (angle_A >= 32 && angle_A <= 35)
{
    if (angle_B >= 25 && angle_B <= 33)
    {
        sb.Append("\\qc \\fs32 Вердикт 5\\par\\par");
        sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append("\\ql \\fs28 Д. Угол наклона опорной
поверхности (Б) равен " + FormatFloat(angle_B) + "\\par");
        sb.Append("\\ql \\fs28 Ацетабулярный
коэффициент равен " + FormatFloat(AK) + "\\par");
        sb.Append("\\ql \\fs28 Показана Г-образная
osteотомия подвздошной кости +ацетабулопластика \\par");
        sb.Append("\\ql \\fs28 За счет остеотомии
подвздошной кости осуществляется наклон впадины во фронтальной

```

```

(латеральный наклон) и сагиттальной (передний наклон) плоскости
\\par");
        sb.Append("\\ql \\fs28 Степень латерального
наклона ацетабулярного фрагмента = " + FormatFloat(FPN) + "\\par");
        sb.Append("\\ql \\fs28 В сагиттальной
плоскости (передний наклон)\\par");
        sb.Append("\\ql \\fs28 Степень переднего
наклона ацетабулярного фрагмента = " + FormatFloat(angle_E) +
"\\par");
        sb.Append("\\ql \\fs28 За счет
ацетабулопластики осуществляется дополнительное латеральное покрытие
головки \\par");
        sb.Append("\\ql \\fs28 Величина трансплантата:
" + FormatFloat(trans) + "\\par");
    }
    else
    {
        sb.Append("\\qc \\fs32 Вердикт 0\\par\\par");
        sb.Append("\\ql \\fs28      " + PHIO +
"\\par");
        sb.Append("\\ql \\fs28      Год рождения - " +
year.ToString() + "\\par\\par");
        sb.Append("\\qc \\fs32 Измеренные параметры
выходят за пределы допустимого диапазона \\par В данном случае
реконструктивная операция не показана\\par");
    }
}
    }
    sb.Append("\\par");
    sb.Append("}");
    return sb.ToString();
}
    public static void PrintVerdict(string PHIO, int
year, string output, string filename)
    {
        try
        {
            File.WriteAllText(filename, output,
Encoding.GetEncoding(1251));
            Console.WriteLine($"Файл успешно сохранен:
{filename}");
        }
        catch (Exception ex)
        {
            Console.Error.WriteLine("Не удалось открыть файл для
записи: " + ex.Message);
        }
    }
    private void btn_complete_Click(object sender, EventArgs e)
    {
        if (textBox_9_2_SDLN.Text == "" || textBox_9_3_PHIO.Text
== "" || textBox_9_4_year.Text == "" || textBox_9_5_age.Text == "")
        {
            MessageBox.Show("Требуемые поля не заполнены!",
"Ошибка",
MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
}

```

```

else
{
    SharedData.PHIO = textBox_9_3_PHIO.Text;
    SharedData.year =
Convert.ToInt32(textBox_9_4_year.Text);
    SharedData.age =
Convert.ToInt32(textBox_9_5_age.Text);
    SharedData.HP = SharedData.Angle_V - 20;
    SharedData.BP = 15 - SharedData.Angle_V;
    SharedData.trans = 15 + SharedData.Dh * (1 -
SharedData.ICAS);
    string output;
    if (SharedData.age >= 13)
    {
        output =
GenerateVerdictFor13AndOlder(SharedData.PHIO,
    SharedData.year,
    SharedData.ICAS,
    SharedData.Angle_A,
    SharedData.Angle_B,
    SharedData.Angle_V,
    SharedData.Angle_Viberg,
    SharedData.Angle_D,
    SharedData.Angle_E,
    SharedData.Dist_D,
    SharedData.Dist_W,
    SharedData.UOB,
    SharedData.Dh,
    SharedData.AK,
    SharedData.SLN,
    SharedData.SPN_1,
    SharedData.HP,
    SharedData.BP,
    SharedData.trans);
    }
    else if (SharedData.age >= 10)
    {
        output = GenerateVerdictFor10To13(SharedData.PHIO,
    SharedData.year,
    SharedData.ICAS,
    SharedData.Angle_A,
    SharedData.Angle_B,
    SharedData.Angle_V,
    SharedData.Angle_Viberg,
    SharedData.Angle_D,
    SharedData.Angle_E,
    SharedData.Dist_D,
    SharedData.Dist_W,
    SharedData.UOB,
    SharedData.Dh,
    SharedData.AK,
    SharedData.SLN,
    SharedData.SPN_1,
    SharedData.HP,
    SharedData.BP,
    SharedData.trans);
    }
}

```

```

    }
    else
    {
        output = GenerateVerdictFor7To9(SharedData.PHIO,
        SharedData.year,
        SharedData.ICAS,
        SharedData.Angle_A,
        SharedData.Angle_B,
        SharedData.Angle_V,
        SharedData.Angle_Viberg,
        SharedData.Angle_D,
        SharedData.Angle_E,
        SharedData.Dist_D,
        SharedData.Dist_W,
        SharedData.UOB,
        SharedData.Dh,
        SharedData.AK,
        SharedData.SLN,
        SharedData.SPN_1,
        SharedData.HP,
        SharedData.BP,
        SharedData.trans);
    }
    string filename = SharedData.PHIO + "
" + SharedData.year.ToString() + ".rtf";
    PrintVerdict(SharedData.PHIO, SharedData.year, output,
filename);
    MessageBox.Show($"Файл {filename} сохранён в папке с
программой.",
        "Информация",
        MessageBoxButtons.OK,
        MessageBoxIcon.Information);
}
}
private void btn_measure_9_2_Click(object sender,
EventArgs e)
{
    if (textBox_9_1_Angle_E.Text == "")
    {
        MessageBox.Show("Угол Е не измерен!",
            "Ошибка",
            MessageBoxButtons.OK,
            MessageBoxIcon.Error);
    }
    else
    {
        SharedData.Angle_E =
Convert.ToDouble(textBox_9_1_Angle_E.Text);
        SharedData.SDLN = SharedData.Angle_B -
SharedData.Angle_E;
        textBox_9_2_SDLN.Text =
Convert.ToString((float)SharedData.SDLN);
    }
}
}
}

```

Приложение В

(обязательное)

Программы автоматизации обработки рентгеновских снимков

Презентация

					ПГУ1.090501.04.001 ПЗ	Лист
						143
Изм.	Лист	№ докум.	Подпись	Дата		



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФГБОУ ВО «ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет вычислительной техники
Кафедра «Вычислительная техника»



Дипломный проект на тему «Разработка программы автоматизации обработки рентгеновских снимков»

Специальность: 09.05.01 – Применение и эксплуатация автоматизированных систем специального назначения

Студент: группы 20ВОЭ1

Быстров Михаил Сергеевич

Руководитель: к.т.н., доцент кафедры ВТ

Бычков Андрей Станиславович

Пенза, 2025

Цель работы

Целью данной работы является создание программы автоматизации обработки рентгеновских снимков и реализация ее на языке программирования C# с использованием технологии Windows Forms

2

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

144

Разработка программы автоматизации обработки рентгеновских снимков

Техническое задание на проектирование

Заказчиками разработки программы автоматизации обработки рентгеновских снимков являются представители Национального медицинского исследовательского центра травматологии и ортопедии им. академика Г.А. Илизарова, г. Курган.

В качестве ТЗ на проектирование ими были представлены подробные словесные описания алгоритмов действий пользователей программы (хирургов, травматологов, ортопедов) при получении информации о заболеваниях опорно-двигательного аппарата человека, в частности, тазобедренного сустава, путем анализа рентгенограмм таза, полученных в разных проекциях.

3

Всего анализу подлежат четыре различные рентгенограммы, три из которых являются обязательными, а четвертая требуется в лишь в определенных исключительных случаях. Необходимость ее применения выясняется в ходе анализа данных первых трех рентгенограмм.

Анализ рентгенограмм включает измерение геометрических параметров суставов, на основании которых делается расчет ряда коэффициентов, характеризующих то или иное заболевание пациента. На основании этого анализа врач выносит вердикт, в котором указывается диагноз заболевания и рекомендации о необходимости или целесообразности проведения той или иной хирургической операции.

В ТЗ приведено словесное описание трех различных алгоритмов действий пользователей для разных возрастных групп пациентов, так как в зависимости от возраста применяются несколько различные подходы при определении диагноза пациента и выработки рекомендаций по его лечению:

- возраст 13 лет и более;
- возраст от 10 до 13 лет;
- возраст от 7 до 9 лет.

4

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

145

Разработка алгоритма работы программы

В результате изучения словесного описания последовательности действий врачей-хирургов при анализе параметров рентгенограмм, были разработаны алгоритмы работы программы автоматизации обработки рентгеновских снимков, расчета и выдачи пациентам соответствующих вердиктов.

Анализ ТЗ показал, что последовательности действий врачей-хирургов при работе с пациентами различных возрастных групп во многом идентичны друг другу. Особенно это касается процессов измерения геометрических параметров суставов по рентгенограммам, тогда как расчетные формулы часто различны и определяются возрастом пациента.

Поэтому на первом этапе разработки алгоритма работы программы была произведена минимизация словесного описания алгоритма путем исключения одинаковых фрагментов для разных возрастных групп.

5

В результате оптимизации в структуре разрабатываемой программы укрупненно можно выделить две основные части (модуля):

- первый модуль - процедура ввода исходных данных о пациенте и параметров рентгенограмм. Этот модуль используется в любом случае, независимо от возраста пациента;

- второй модуль - процедура обработки введенной информации, формирования вердикта и выдачи его на печать (в текстовый файл для дальнейшей распечатки) или на экран. Данный модуль состоит из трех функционально законченных фрагментов программы (функций) для соответствующих возрастных групп, так как в зависимости от возраста применяются несколько различные подходы при определении диагноза пациента и выработки рекомендаций по его лечению.

Для пациентов в возрасте до 7 лет данная методика диагностики и лечения, а соответственно, данная программа неприменимы.

6

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

146

Разработка процедуры ввода исходных данных о пациенте, измерения и расчета параметров рентгенограммы

Первый модуль, процедура ввода исходных данных о пациенте, измерения и расчета параметров рентгенограммы, применяется для всех возрастных групп.

На выходе данной процедуры, помимо информации о фамилии, имени, отчестве и возрасте пациента, имеем ряд переменных, содержащих результаты измерений и расчета требуемых параметров, необходимых для вынесения вердикта:

- Dh (диаметр головки бедра);
- Da (диаметр ацетабулярного фрагмента бедра);
- R (расстояние между фигурами слезы);
- ISh (индекс сферичности головки бедра);
- Dist_A, Dist_B, Dist_D, Dist_W (различные измеряемые расстояния);
- Okano (индекс Окано);

7

- Angle_A, Angle_B, Angle_V, Angle_E (измеряемые углы);
- ISA (индекс сферичности ацетабулярного фрагмента);
- ICAS (индекс конгруэнтности или соответствия);
- AK (ацетабулярный коэффициент);
- UOB, SLN, SPN_1, SDLN, HP, BP, trans (дополнительные параметры);
- PHIO (ФИО);
- year (год рождения пациента);
- age (возраст пациента, количество полных лет).

Все вышеперечисленные переменные являются исходными данными для второго модуля – модуля обработки введенной информации, формирования и выдачи вердикта.

Схема алгоритма первого модуля приведена на рисунке 1.

Схема алгоритма второго модуля приведена на рисунке 2.

На схемах наглядно представлена последовательность операций, выполняемых пользователем при вводе исходных данных с рентгеновских снимков и при ее обработке.

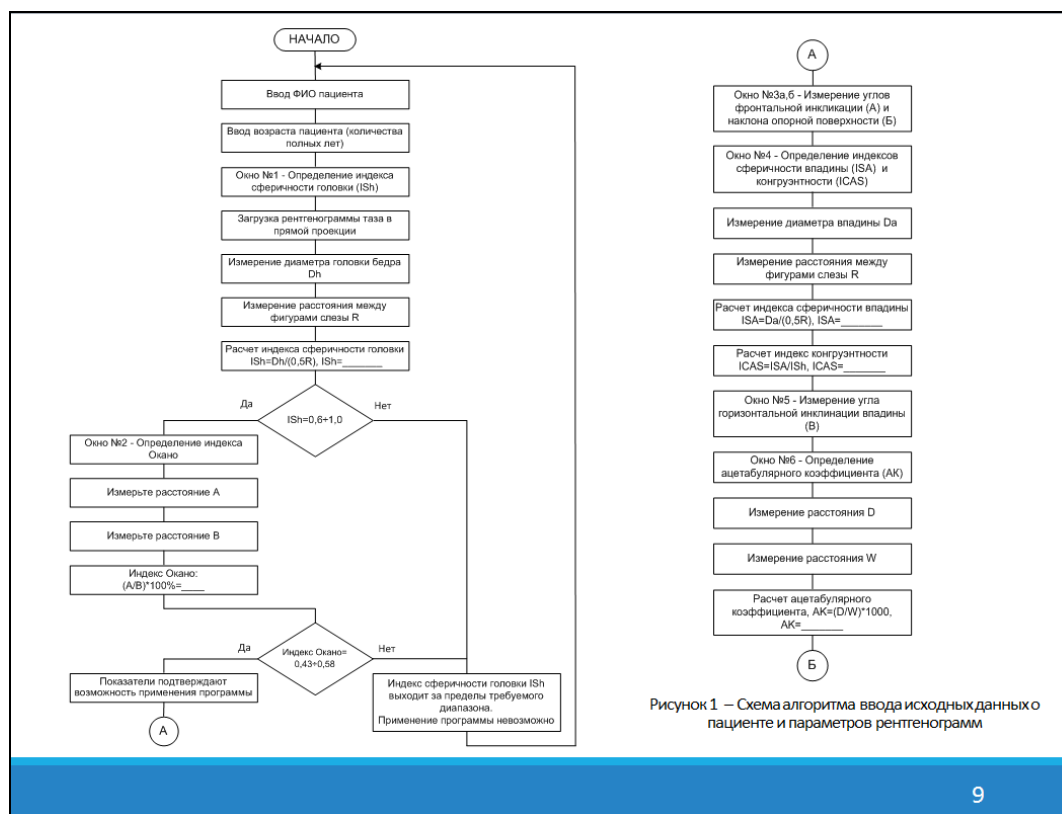
8

Изм.	Лист	№ докум.	Подпись	Дата

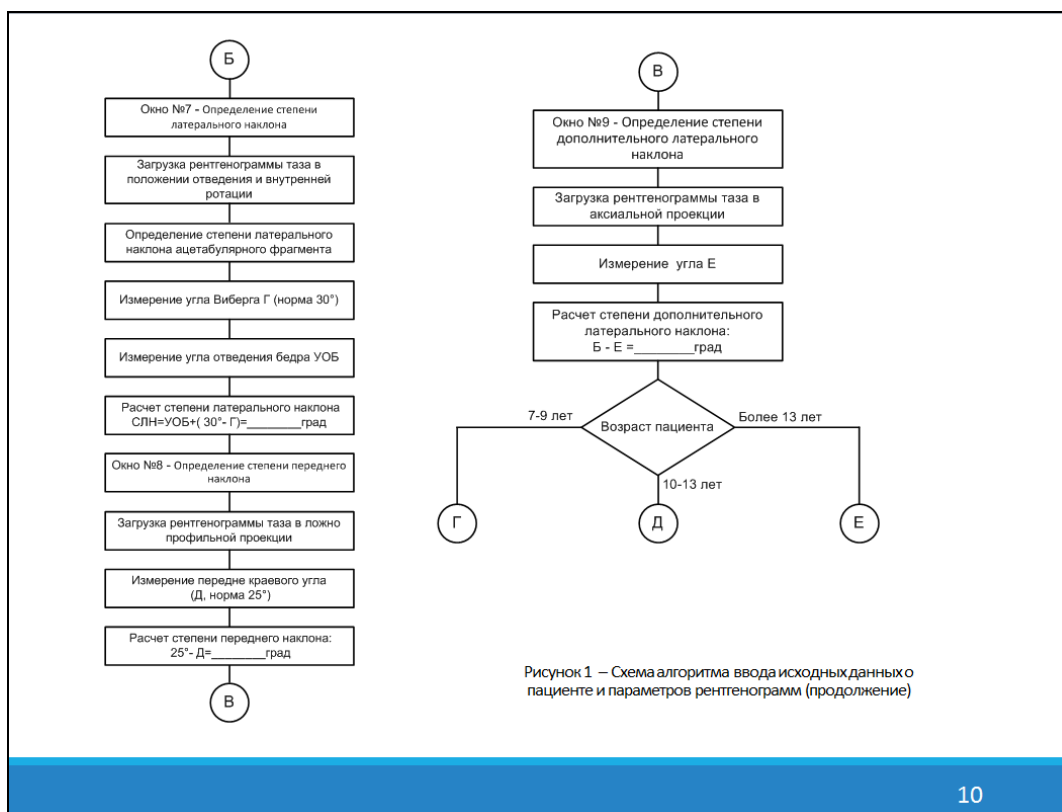
ПГУ1.090501.04.001 ПЗ

Лист

147



9



10

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

148

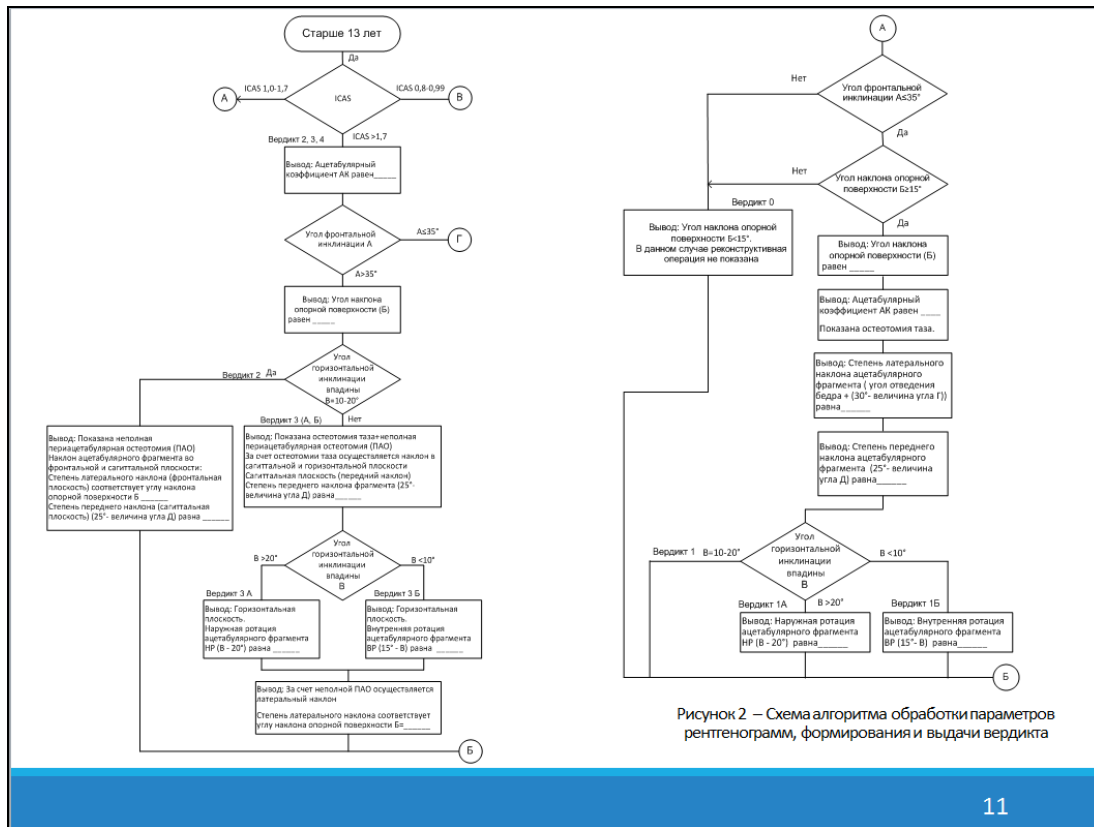


Рисунок 2 – Схема алгоритма обработки параметров рентгенограмм, формирования и выдачи вердикта

11

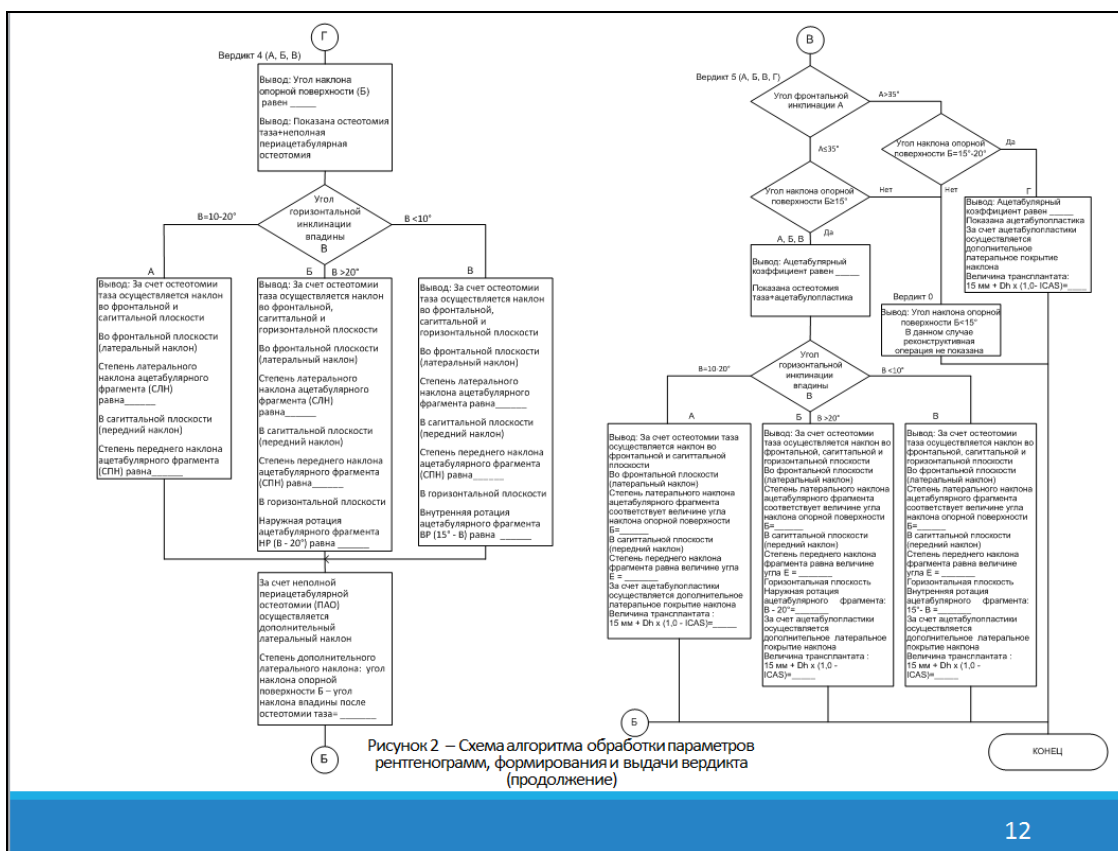


Рисунок 2 – Схема алгоритма обработки параметров рентгенограмм, формирования и выдачи вердикта (продолжение)

12

Разработка экранных форм для ввода параметров рентгенограмм

В соответствии с ТЗ, в ходе дипломного проектирования были разработаны девять экранных форм, предназначенных для ввода исходных параметров рентгенограмм пациента, по которым осуществляется определение расчетных параметров анализируемого участка опорно-двигательного аппарата и вынесение соответствующего вердикта, содержащего диагноз заболевания пациента и рекомендации по его дальнейшему лечению, а также для управления процессами расчета параметров рентгенограмм.

Все экранные формы имеют схожую структуру, выполнены в едином стиле, имеют примерно одинаковый состав органов управления и отображения информации.

Функции управляющих элементов просты и интуитивно понятны пользователю. Примеры экранных форм приведены ниже.

13

Окно №1 - Загрузка рентгенограммы №1 и определение индекса сферичности головки ISh

Рентгенограмма 1
Шпателька 1
Выход

Загрузите рентгенограмму таза в прямой проекции

Выберите файл

Определение индекса сферичности головки ISh

Помощь

Измерьте диаметр головки Dh

Dh=_____мм

Измерьте расстояние между фигурами слезы R

R=_____мм

Рассчитать

Индекс сферичности головки $ISh = Dh / (0,5R)$, $ISh =$ _____

Далее

Индекс сферичности головки (ISh) - отношение диаметра окружности, соответствующей форме головки (Dh) к половине расстояния между фигурами слезы (R)

14

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

150

Окно №7 - Загрузка рентгенограммы №2 и определение степени латерального наклона

Загрузите рентгенограмму таза в положении отведения и внутренней ротации

Определение степени латерального наклона ацетабулярного фрагмента

Измерьте угол Виберга Г (норма 30°)

Измерьте угол отведения бедра УОБ

Степень латерального наклона (СЛН): $УОБ + (30^\circ - Г) =$ _____ град

Выберите файл

Помощь

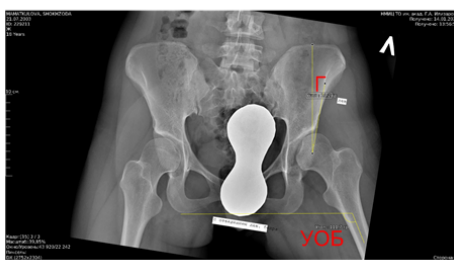
Г= _____ град

УОБ= _____ град

Рассчитать

Далее

Рентгенограмма 2 Шпаргалка 8 Выход



Угол Виберга (Г) – угол между линией проведенной через центр головки касательно к наружному краю впадины и вертикалью, проведенной через центр головки

Угол отведения бедра (УОБ) – угол между горизонтальной линией, проведенной через нижний край зоны склероза (зона нагрузки) и осевой линией бедра в положении отведения и внутренней ротации

Определение степени латерального наклона - определяется угол Виберга Г (норма 30°). Степень латерального наклона ацетабулярного фрагмента - угол отведения + (30° - величина угла Г)

15

Окно №8 - Загрузка рентгенограммы №3 и определение степени переднего наклона

Загрузите рентгенограмму таза в ложно профильной проекции

Определение степени переднего наклона

Измерьте передний краевой угол (Д, норма 25°)

Степень переднего наклона: $25^\circ - Д =$ _____ град

Выберите файл

Помощь

Д= _____ град

Рассчитать

Далее

Рентгенограмма 3 Шпаргалка 9 Выход



Угол Д – угол между линией проведенной через центр головки касательно к переднему краю впадины и вертикалью, проведенной через центр головки

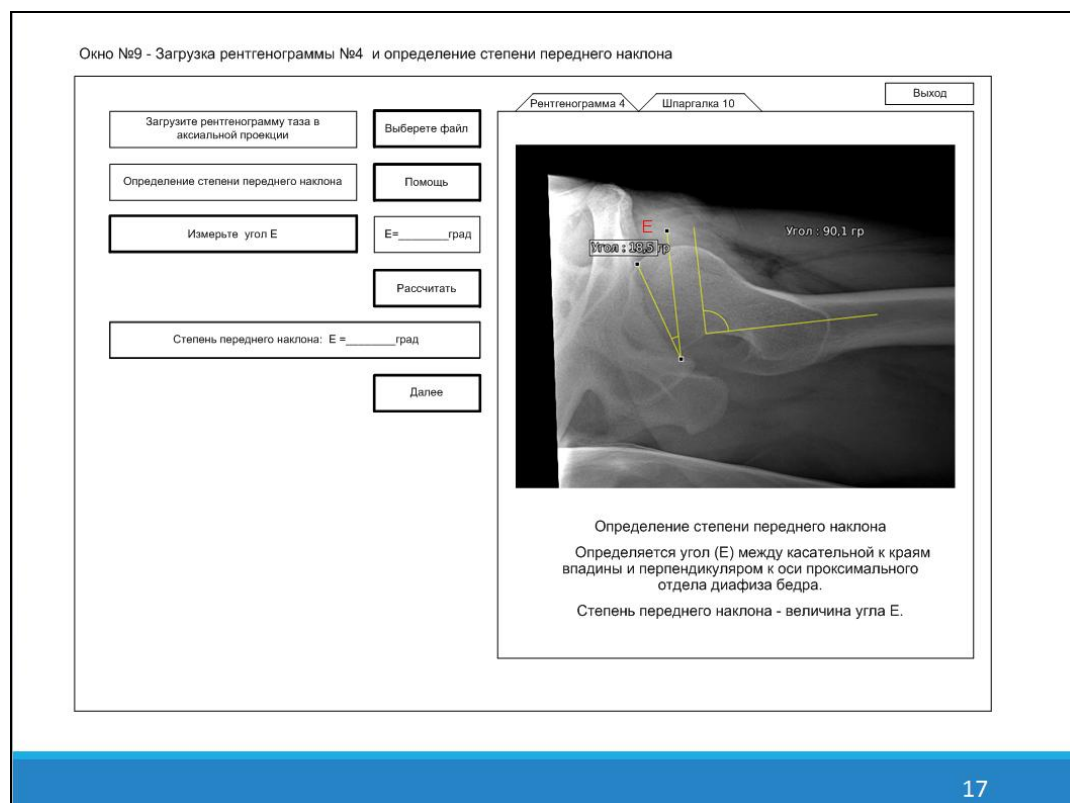
16

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

151



Реализация программы автоматизации обработки рентгеновских снимков на языке высокого уровня

Выбор языка программирования и среды разработки

Выбор языка программирования C# и платформы .NET обусловлен рядом преимуществ. C# - это мощный объектно-ориентированный язык, в который входят множество библиотек, включая библиотеки для работы с графикой. Это значительно упрощает создание приложений с графическими интерфейсами.

Интеграция со средой разработки Visual Studio позволяет быстро и легко как разрабатывать новые программные продукты, так и тестировать. Windows Forms была выбрана как проверенная и достаточно быстрая платформа для создания настольных приложений, работающих на Windows. Она проста в освоении, позволяет быстро создавать прототипы интерфейсов и хорошо интегрируется с C#, что позволяет разработчику сосредоточиться на логике приложения.

18

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

152

Описание программы автоматизации обработки рентгеновских снимков

Программа «Medic» — это Windows Forms приложение, разработанное на C# для анализа рентгенограмм таза. Оно предназначено для загрузки изображений рентгенограмм, выполнения измерений различных параметров (углов, расстояний, диаметров) и генерации вердикта о необходимости хирургического вмешательства на основе этих данных. Программа имеет модульную структуру, состоящую из нескольких окон (форм), каждое из которых отвечает за определённый этап анализа, а также вспомогательных файлов для управления данными и интерфейсом.

Общая структура программы

Программа состоит из следующих основных файлов:

- Program.cs — точка входа в приложение;
- SharedData.cs — статический класс для хранения общих данных;
- Window_(1-9).cs — файлы с логикой для каждого окна программы;
- Window_(1-9).Designer.cs — файлы с описанием отображения элементов интерфейса.

Каждое окно (Window_1–Window_9) представляет собой отдельный этап анализа рентгенограмм, где пользователь загружает изображения, выполняет измерения и переходит к следующему этапу.

Данные между окнами передаются через класс SharedData.

19

Тестирование программы

За неимением реальных рентгенограмм были использованы отредактированные рисунки с «шпаргалок», в которых отсутствует текст подсказок, а также указания наименований длин и углов, измеряемых величин и некоторые измерительные линии. На рисунке 3, приведенном ниже, в программу была загружена первая рентгенограмма:

20

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

153

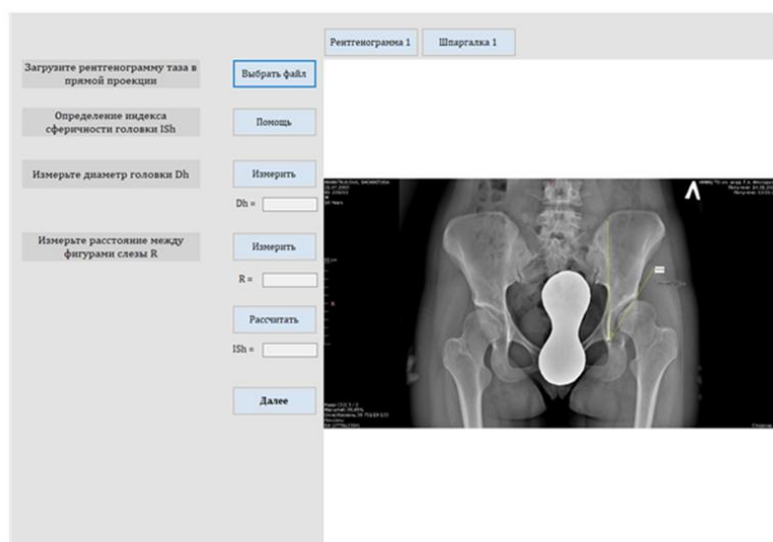


Рисунок 3 – Окно программы после открытия первой рентгенограммы
Далее был выполнен переход к «шпаргалке» при помощи нажатия кнопки «Шпаргалка 1» (см. рисунок 4).

21

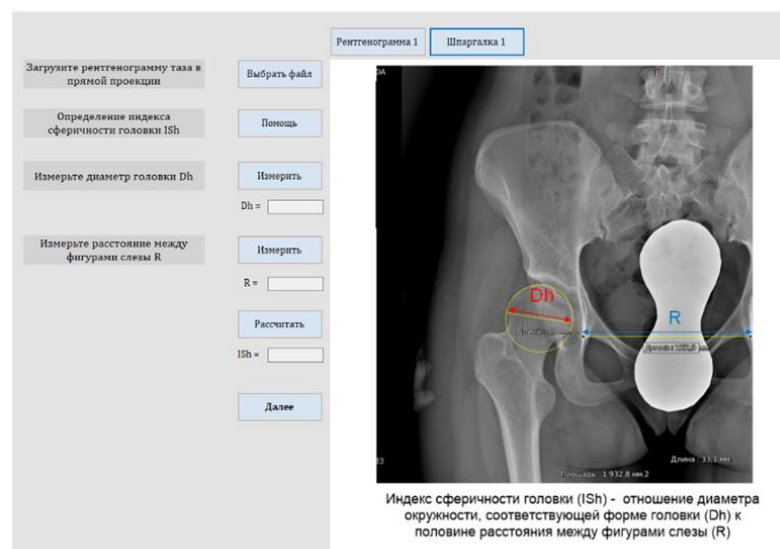


Рисунок 4 – Окно программы после открытия первой шпаргалки
После изучения информации был выполнен обратный переход при помощи кнопки «Рентгенограмма 1» и были выполнены соответствующие измерения и расчёт ISh (см. рисунок 5).

22

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

154

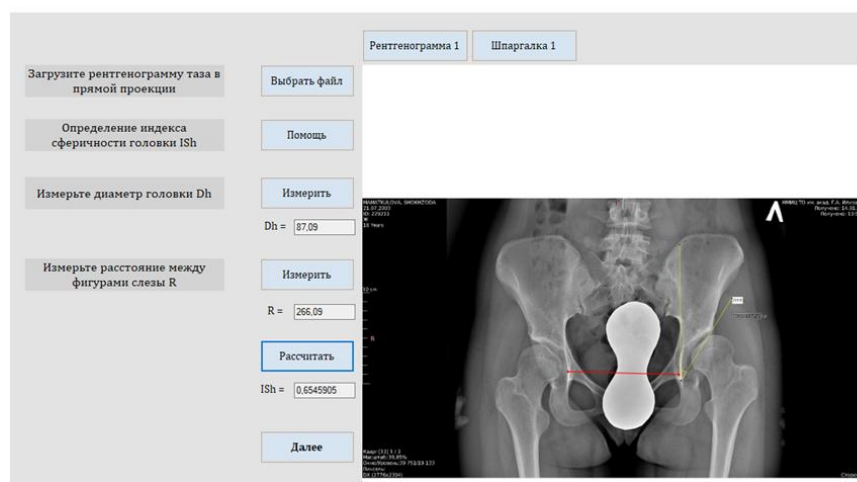


Рисунок 5 – Окно программы после измерения и расчёта ISh
После измерений был осуществлён переход на следующий экран, где была открыта очередная «шпартгалка» (см. рисунок 6).

23

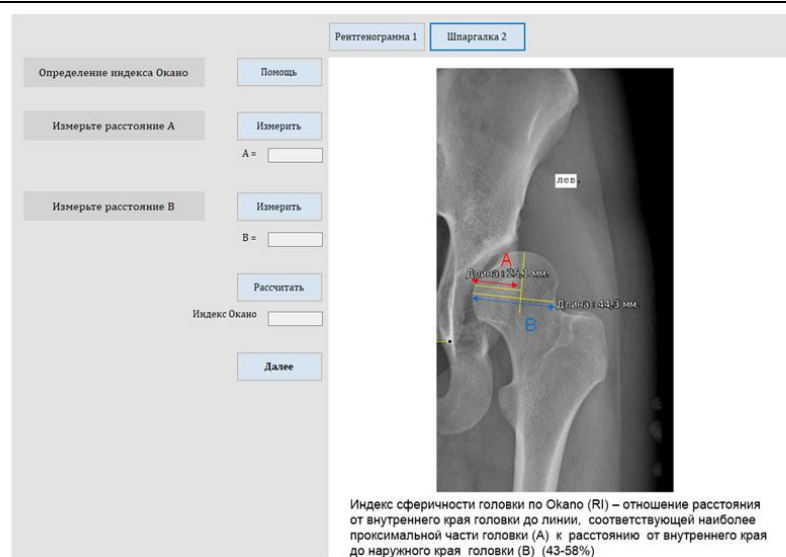


Рисунок 6 – Окно программы после открытия второй шпартгалки
После изучения информации был осуществлён переход обратно к рентгенограмме и были выполнены соответствующие измерения и расчёт индекса Окано. Был произведен переход на следующий экран (см. рисунок 7).

24

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

155

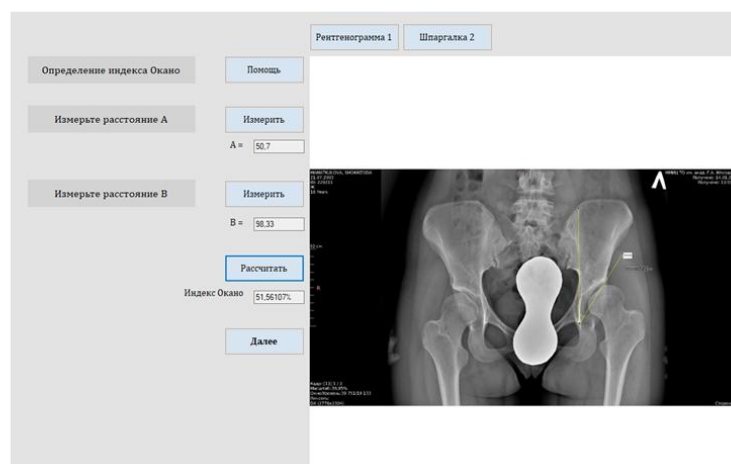


Рисунок 7 – Окно программы после измерения и расчёта индекса Окано

25

Дальнейшая работа с программой выполняется аналогично. При переходе к последней операции по измерению и расчету необходимых параметров рентгенограмм вместо кнопки «Продолжить» появляется кнопка «Завершить». см. рисунок 8. Здесь же появляются поля для ввода персональных данных пациента: «ФИО:», «Год рождения» и «Количество полных лет:».

При нажатии на кнопку «Завершить», формируется вердикт для данного пациента и записывается в текстовый файл формата rtf. При этом появляется информирующее окно, показанное на рисунке 9.

Ниже, на рисунке 10, приведен файл «Иван Иванович Иванов 2007.rtf» с вердиктом, сформированный и записанный в ту же папку, где располагается программа.

Как видно из рисунка 10, программа успешно вывела результат в файл, следовательно, она работает исправно и в дальнейшем её можно применять для автоматизации обработки рентгеновских снимков.

26

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

156

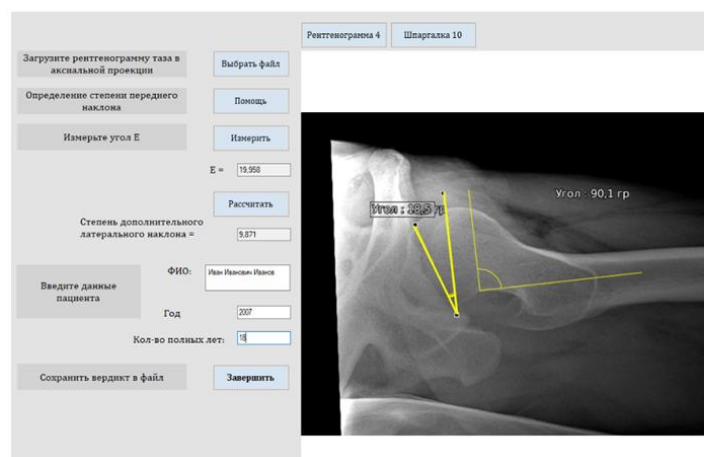


Рисунок 8 – Окно программы после измерений, расчёта и заполненными полями с данными пациента

27

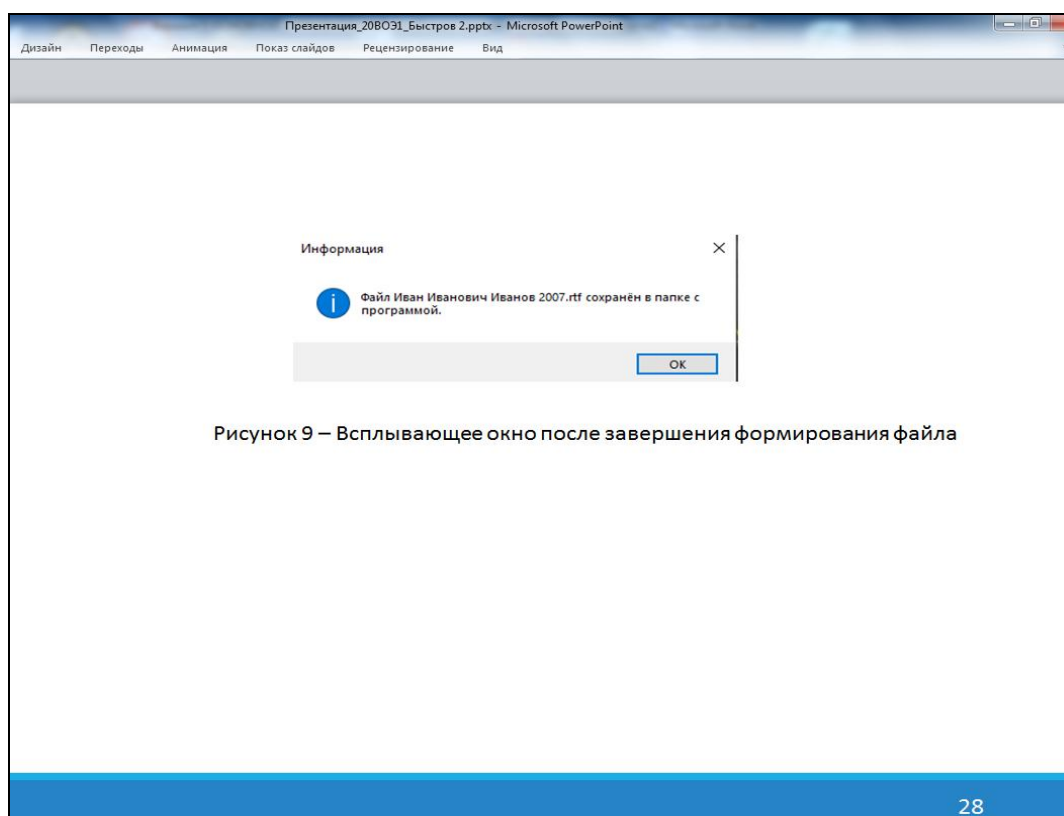


Рисунок 9 – Всплывающее окно после завершения формирования файла

28

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

157

Вердикт 4

Иван Иванович Иванов

Год рождения - 2007

Ацетабулярный коэффициент АК равен 267,62

Угол наклона опорной поверхности (Б) равен 29,83

Показана остеотомия таза + неполная периацетабулярная остеотомия

Б. За счет остеотомии таза осуществляется наклон во фронтальной, сагиттальной и горизонтальной плоскости

Во фронтальной плоскости (латеральный наклон):

Степень латерального наклона ацетабулярного фрагмента (СЛН) равна 132,17

В сагиттальной плоскости (передний наклон):

Степень переднего наклона ацетабулярного фрагмента (СПН) равна 21,01

В горизонтальной плоскости:

Наружная ротация ацетабулярного фрагмента НР равна 5,43

За счет неполной периацетабулярной остеотомии (ПАО) осуществляется дополнительный латеральный наклон

Степень дополнительного латерального наклона = 9,87

Рисунок 10 – Файл вердикта «Иван Иванович Иванов 2007.rtf»

29

Спасибо за внимание!

30

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.090501.04.001 ПЗ

Лист

158