ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК
КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

АВТОМАТИЧЕСКАЯ СИСТЕМА АНАЛИЗА ИЗОБРАЖЕНИЯ ГРАФИКА ЭКГ С ЦЕЛЬЮ ОПРЕДЕЛЕНИЯ ЧСС

Выпускная квалификационная работа

обучающегося по направлению подготовки
02.04.01 Математика и компьютерные науки
очной формы обучения,
группы 07001631
Пеньковой Анны Евгеньевны

Научный руководитель
К.т.н., доцент Муромцев В.В.

Рецензент:
К.т.н., доцент Зайцева Т.В.

БЕЛГОРОД 2018

Содержание

ВВЕДЕНИЕ

В постоянно меняющемся мире непрерывно обновляются технологии во всех сферах жизни человека. К медицинскому оборудованию требования особенно строги. С постоянным совершенствованием аппаратуры и программного обеспечения всё точнее диагностика, всё шире сфера применения электронно-вычислительной техники в медицине.

Научные достижения миллионов специалистов по всей планете привели нынешнюю ситуацию к невероятным результатам. Автоматический анализ биометрической информации позволяет свести к минимуму вероятность врачебных ошибок, делая жизнь людей более качественной и полноценной.

Современное медицинское оборудование если не все 100%, то на 80-это компьютер. С процессором, памятью, специфической архитектурой, укомплектованной под конкретные виды обследований.

На сегодняшний день биомедицинские сигналы, снятые с пациента, представляют собой информацию, записанную в электронном виде. В этом случае, появляется возможность провести более качественный анализ, нежели при информации, представленной «на бумаге». А также, данные о сигналах, представленные в электронном виде, можно обработать при помощи различных аппаратных и программных средств, которые позволят нам дать точную оценку о нарушениях, патологиях и анормальностях. Разработка программы для анализа электрокардиосигналов (ЭКС) сегодня является достаточно актуальной, так как анализ сигналов один из важных этапов в компьютерной диагностике, а также создание новых программ позволит ставить наиболее точные диагнозы и улучшить качество диагностики сердечных заболеваний.

Данная работа посвящена автоматизации анализа ЭКГ. Проекция объёмных электрических процессов происходящих в сердце и наблюдаемых на поверхности тела несёт информацию о состоянии сердечно-сосудистой системы, о патологиях и возрастных изменениях.

В основу работы положено изучение и разработка теоретических и практических методов создания нового поколения алгоритмов длительного и непрерывного анализа изображения ЭКГ. С этой целью использованы наиболее современные подходы к обработке сигналов, отвечающие современному уровню требований к качеству.

Анализ изображений электрокардиограммы является объектом исследования данной работы.

За предмет исследования взяты алгоритмы определения состояния сердечно-сосудистой системы.

Использование разработанной методики в медицинском оборудовании повысит точность и надёжность диагностики, что усилит эффективность лечебных процедур. Увеличит качество медицинской помощи в борьбе с сердечно-сосудистыми патологиями.

Цель работы: Создание алгоритмов и основанного на них программного обеспечения, позволяющего более детально и точно диагностировать патологии сердечно-сосудистой системы.

Задачи исследования, призванные достичь поставленной цели:

1. Анализ предметной области.

2. Обзор современного рынка программного обеспечения.

3. Аналитическая обработка современных методов и программ, используемых в диагностической аппаратуре для определения патологий сердечно-сосудистой системы человека.

4. Создание метода, процедуры и алгоритма анализа изображений ЭКГ.

5. Формирование новейшего программно-аналитического ресурса, обеспечивающего высокоточные и на порядок более надёжного

автоматического анализа изображений ЭКГ с целью получения значимой для диагностики информации о состоянии сердечно-сосудистой системы.

Для получения результата в поставленных задачах использовались методы математического моделирования, способы программного распознавания образов, анализа процессов, сгенерированных датчиками случайных чисел.

Опытные исследования выполнялись при использовании наборов реальных записей сигналов, полученных в медучреждениях. Также для сравнения брались данные, полученные отечественными и зарубежными исследователями. ПО для постановки экспериментов разрабатывалось в среде MS Visual C++.

Достоверность научных положений и выводов подтверждается результатами использования математических методов анализа, результатами экспериментов с использованием как модельных, так и реальных сигналов, оценкой эффективности разработанных алгоритмов и методов, а также результатами практического использования созданных алгоритмических и программных средств.

Пояснительная записка состоит из введения, трех разделов, заключения, списка используемых библиографических источников и приложений.

Во введении рассматривается актуальность выбранной темы, выбираются объект и предмет исследования, формулируется цель и ставятся задачи. Также, во введении представлена новизна исследования, применяемые методы исследования и структура магистерской диссертации.

В первом разделе приведено аналитическое исследование предметной области, включая анализ современного состояния как методов, так и аппаратно-программного обеспечения обработки графиков ЭКС.

Во втором разделе приведена разработка усовершенствованных и модифицированных алгоритмов и процедур анализа изображений ЭКГ.

В третьем разделе представлена программная реализация разработанных алгоритмов и процедур, включая проведение вычислительных экспериментов на реальных данных.

В заключении приведены основные выводы и результаты.

Пояснительная записка написана на 79 стр., включает 22 рисунков, 3 таблиц и 1 приложений.

1. АНАЛИТИЧЕСКОЕ ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Основные положения предметной области

Электрокардиограмма регистрирует только электрические процессы в миокарде: деполяризацию (возбуждение) и реполяризацию (восстановление) клеток миокарда, как показано на рис 1.1.

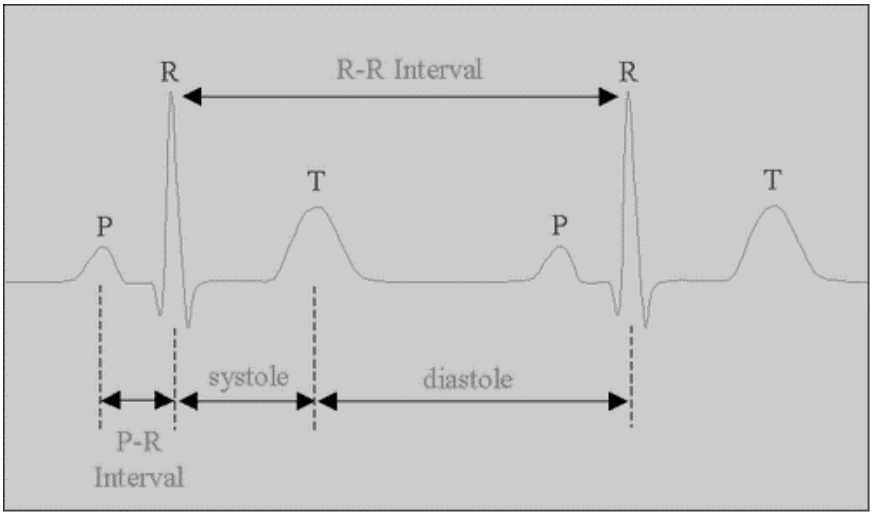

Рисунок 1.1. Соотношение интервалов ЭКГ с фазами сердечного цикла (систола и диастола желудочков)

При функционировании мышечной клетки без отклонений деполяризация способствует сокращению, а реполяризация к расслаблению мышц. Для более простого понимания можно принять в использование оборот «сокращение-расслабление» вместо «деполяризация-реполяризация», однако, это не вполне правильно, учитывая то, что есть определение «электромеханическая диссоциация», при которой деполяризация и реполяризация могут не привести к видимому сокращению и расслаблению миокарда.

Для исследования и дешифровки ЭКГ, следует разобрать из каких структурных составляющих она складывается.

Каждая ЭКГ имеет в собственном составе зубцы, сегменты и интервалы, которые представлены на рис. 1.2.

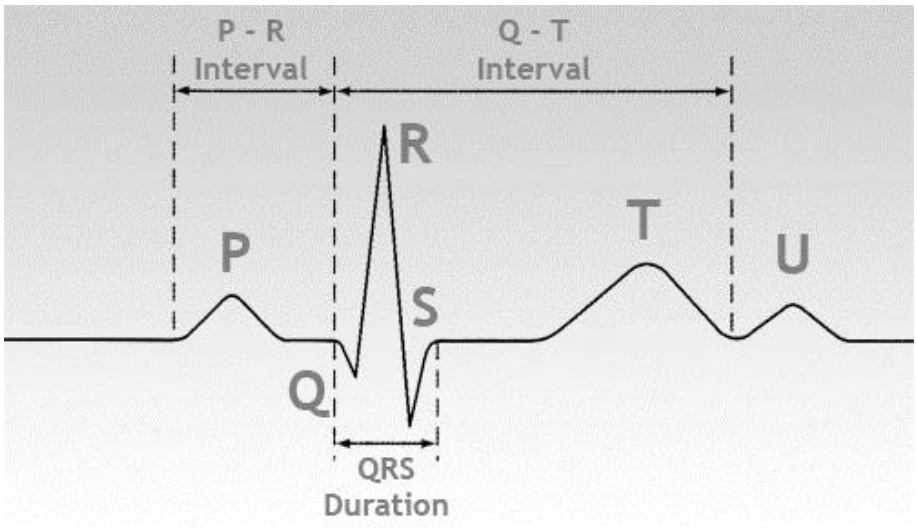

Рисунок 1.2 Зубцы и интервалы на ЭКГ.

Зубцы – это резко и четко обозначенные выпуклости и вогнутости на электрокардиограмме.

Следует отмечать следующие зубцы:

- P (сокращение предсердий),
- Q, R, S (все 3 зубца определяют сокращение желудочков),
- T (расслабление желудочков),
- U (неустойчивый зубец, определяется редко).

Отрывок прямой линии (изолинии), который проходит от одного зубца до следующего принято называть сегментом. Подобные сегменты, такие как P-Q и S-T, определяют наибольшую важность. К примеру, сегмент P-Q проявляется на основании отсрочки осуществления возбуждения в предсердно-желудочковом (AV-) узле.

Группа, состоящая из нескольких зубцов, сегментов или зубцов и сегментов вкупе, называется интервалом. Его можно увидеть на рис 1.3. Подобным образом, интервал равняется сумме зубца и сегмента. Наиболее весомыми представляются интервалы P-Q и Q-T.

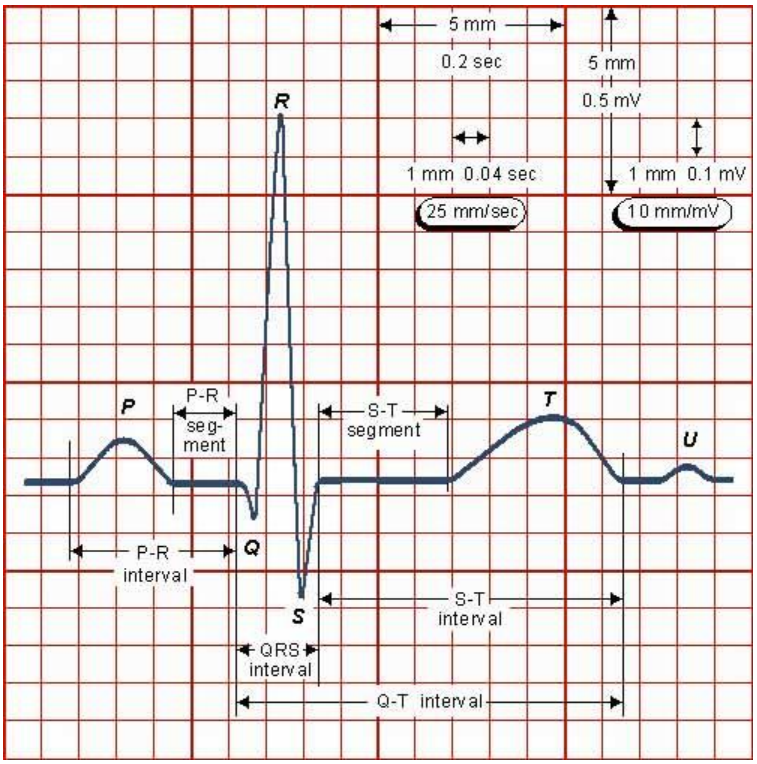

Рисунок 1.3. Зубцы, сегменты и интервалы на ЭКГ.

Миокард желудочков имея значительную межжелудочковую перегородку более массивнее миокарда предсердий. Поэтому распространение возбуждения в нем обуславливается появлением характерного и сложного комплекса QRS зубцов на ЭКГ. Рассмотрим некоторые зубцы данного комплекса.

В заданном комплексе может быть определены некоторые группы зубцов.

Изначально обратим внимание на амплитуду (размеры) отдельных зубцов комплекса QRS. Зубец превышающий амплитуду 5 мм, обозначают заглавной (большой) буквой Q, R или S.Зубец с амплитудой менее 5 мм, обозначают строчной (маленькой) буквой: q, r или s.

Если образовывается зубец, который имеет амплитуду менее пяти миллиметров, то его обозначают строчной буквой: q, r или s.

Любой R(r) зубец QRS, который можно увидеть на рис 1.4, комплекса направленный вверх называют положительным. В случае если зубцов некоторое количество, тогда все следующие за этим зубцы обозначаются с помощью штрихования: R, R’, R” и далее, в зависимости от количества зубцов. Отрицательный, то есть направленный вниз зубец группы QRS, который находится перед основным зубцом R, обычно маркируется как Q (q), а после — как S (s). В случае если комплекс QRS не содержит в своем составе положительных зубцов, то желудочковый комплекс обозначается как QS.

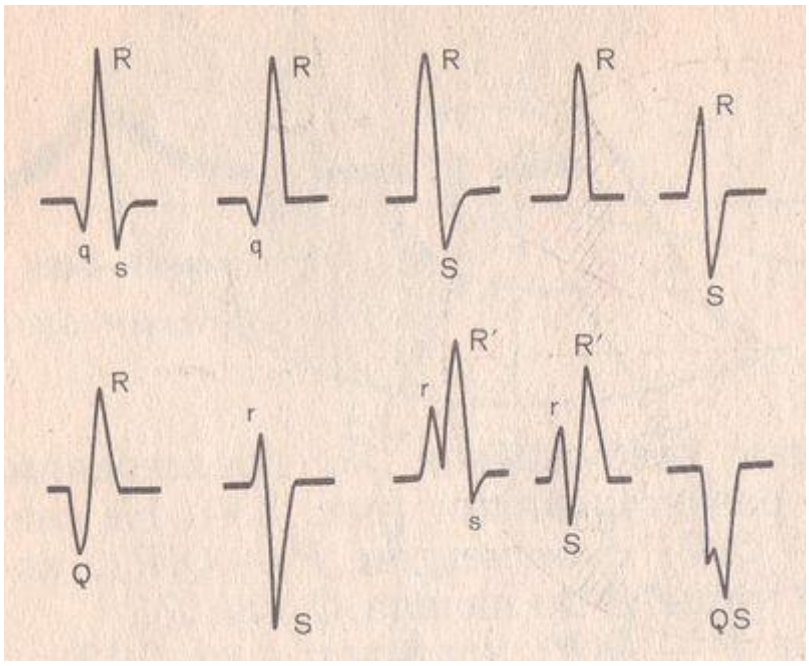

Рисунок 1.4. Варианты комплекса QRS.

Как правило зубец Q характеризует деполяризацию межжелудочковой перегородки, зубец R — ведущего веса миокарда желудочков, зубец S — базальных (то есть около предсердий) отделов межжелудочковой перегородки. Зубец $R_{V1, V2}$ показывает возбуждение межжелудочковой стенки, а $R_{V4, V5, V6}$ — раздражение мышцы левого и правого желудочков. Атрофия части миокарда (как например, во время инфаркта) обуславливает экспансию зубца Q, именно поэтому на данный зубец всегда обращают особое внимание

(подробнее — в 3-й части цикла). На рис. 1.5 показана нормальная электрокардиограмма здорового человека.

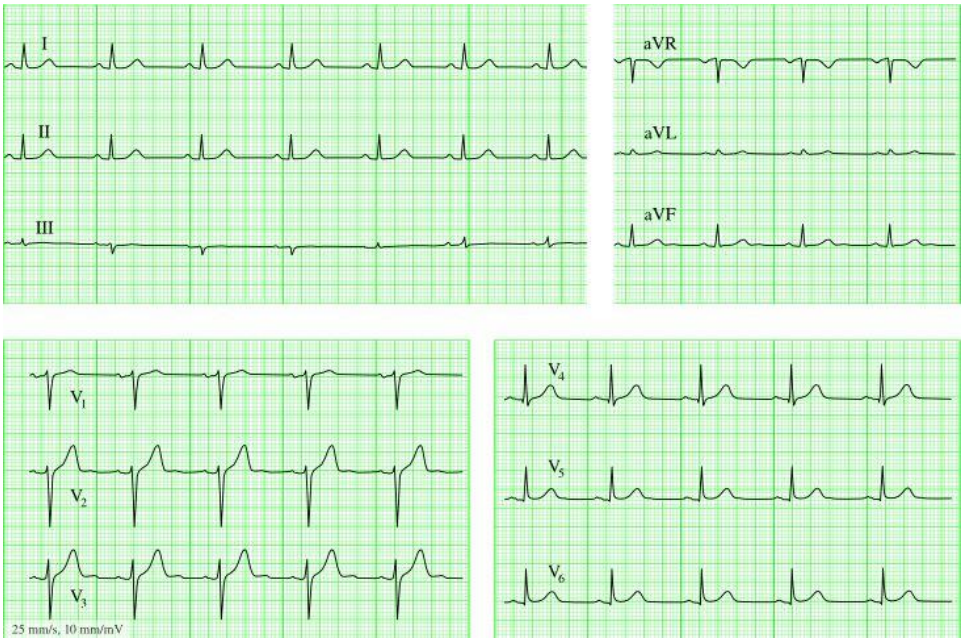

Рисунок 1.5 Нормальная электрокардиограмма.

Как правило, расшифровка ЭКГ совершается следующим образом (общая блок-схема приведена на рис. 1.6):

1. Сверка точности регистрации ЭКГ.
2. Оценка сердечного ритма и проводимости:
   - Анализ регулярности сердечных сокращений,
   - Калькуляция частот сердечных сокращений (ЧСС),
   - Идентификация источника возбуждения,
   - Оценивание проводимости.
3. Установление электрической оси сердца.
4. Оценка предсердного зубца P и интервала P - Q.
5. Оценка желудочкового комплекса QRST:
   - Оценка комплекса QRS,
   - Оценка сегмента RS - T,
   - Оценка зубца T,
   - Оценка интервала Q - T.
6. Электрокардиографическое заключение.

Рисунок 1.6 Объединенная схема расшифровки ЭКГ

1) Сверка точности регистрации ЭКГ

На старте каждой ЭКГ-ленты обязан иметься калибровочный сигнал называемый контрольным милливольтом. Для его получения в начале записи подается стандартное напряжение в 1 милливольт, которое должно показать на ленте отклонение в 10 миллиметров, как показано на рис 1.7. Если калибровочный сигнал отсутствует, запись ЭКГ не может считаться

правильной. В нормальных условиях, по крайней мере одно из стандартных или усиленных отведений от конечностей, амплитуда обязана превышать 5 миллиметров, а в грудных отведениях — 8 миллиметров. Когда амплитуда ниже заданного показателя, это называется сниженным вольтажом ЭКГ, который может встречаться лишь при некоторых патологических состояниях.

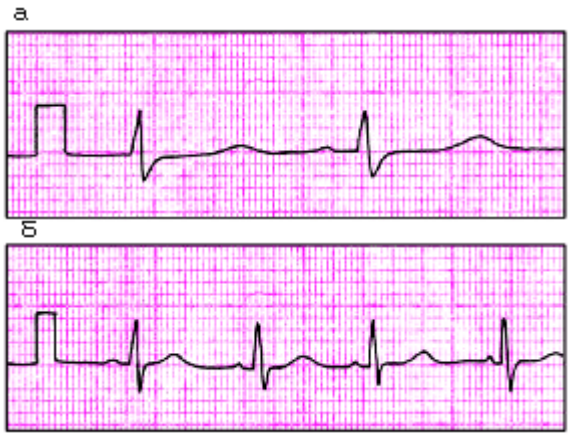

Рисунок 1.7 Контрольный милливольт на ЭКГ (в начале записи).

2) Оценка сердечного ритма и проводимости происходит по следующим этапам:

a. Анализ регулярности сердечных сокращений

Анализ регулярности ритма производится по интервалам R-R. Если зубцы располагаются на одинаковом расстоянии друг от друга, ритм будет называться регулярным, или правильным. Допустим разброс длительности некоторых отдельно взятых интервалов R-R не более ± 10 процентов от среднего времени длительности. Когда ритм синусовый, он как правило является правильным.

b. Калькуляция частоты сердечных сокращений (ЧСС)

На ЭКГ-ленте всегда напечатаны большие и маленькие квадраты. Каждый из больших квадратов включает в себя 25 маленьких квадратов (5 по вертикали и 5 по горизонтали). При ускоренном подсчете ЧСС, ксли ритм правильный, то считают число больших квадратов между двумя соседними зубцами R - R.

При скорости ленты в 50 мм/с: ЧСС = 600 / (количество больших квадратов).

При скорости ленты в 25 мм/с: ЧСС = 300 / (количество больших квадратов).

На иллюстрации ЭКГ, расположенной, выше интервал R-R равняется примерно 4.8 больших клеток, что при скорости записи в 25 мм/с дает 300 / 4.8 = 62.5 уд./мин.

На скорости в 25 мм/с каждая маленькая клетка равняется 0.04 с, а на скорости в 50 мм/с — 0.02 с. Это используется для распознавания длительности зубцов и интервалов на ЭКГ.

При наличии аритмии, обычно, подсчитывают максимальную и минимальную частоту сердечных сокращений, согласно длительности наименьшего и наибольшего интервала R-R соответственно.

с. Идентификация источника возбуждения

Иными словами, необходим для поиска водителя ритма, который провоцирует сокращения предсердий и желудочков. Часто бывает, что это становится одним из самых сложных этапов, из-за того, что различные отклонения возбудимости и проводимости могут крайне запутанно сочетаться, что, в свою очередь, имеет шансы привести к неправильной постановке диагноза и неправильному лечению. Для того, чтобы правильно идентифицировать источник раздражения на ЭКГ, нужно хорошо знать проводящую систему сердца.

Синусовый ритм, показанный на рис 1.8 является нормальным ритмом, в то время как все остальные ритмы являются патологическими.

Очаг возбуждения локализируется в синусно-предсердном узле. Признаки на ЭКГ:

- Во втором стандартном отведении зубцы Р всегда строго положительные и находятся перед каждым комплексом QRS;
- Зубцы Р в одном и том же отведении всегда имеют постоянную одинаковую форму.

Рисунок 1.8 Зубец Р в синусовом ритме.

Предсердный ритм.

Иллюстрация на 1.9. Когда источник раздражения находится в нижних секторах предсердий, волна возбуждения направляется на предсердия снизу вверх (ретроградно), поэтому:

- Во втором и третьем отведениях зубцы Р отрицательные;
- Зубцы Р есть перед каждым комплексом QRS.

Рисунок 1.9 Зубец Р в предсердном ритме.

Ритмы из АВ-соединения.

В случаях, когда водитель ритма находится в атрио-вентрикулярном (предсердно-желудочковом узле) узле, желудочки возбуждаются как обычно (сверху вниз), а предсердия - ретроградно (т.е. Снизу вверх). При этом на ЭКГ отображается следующее:

- Зубцы Р могут отсутствовать совсем, так как наслаиваются на нормальные комплексы QRS, как показано на рис. 1.10;
- Зубцы Р могут быть отрицательными, и располагаться после комплекса QRS, как показано на рис. 1.11.

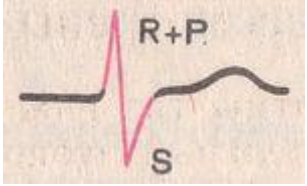

Рисунок 1.10 Ритм из AV-соединения и наложение зубца Р на комплекс QRS.

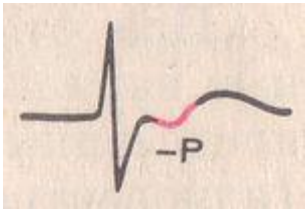

Рисунок 1.11 Ритм из AV-соединения, зубец P находится после комплекса QRS.

Частота сердечных сокращений при ритме из АВ-соединения гораздо меньше синусового ритма и равняется примерно 40-60 ударов в минуту.

Желудочковый, или идиовентрикулярный, ритм, показанный на рис 1.12.

В данном случае источник ритма ─ это проводящая система желудочков. Раздражение распространяется по желудочкам неверными путями и, как следствие, медленнее. Особенности идиовентрикулярного ритма следующие:

- Комплексы QRS расширены и деформированы (выглядят “страшновато”). В норме длительность комплекса QRS равна 0.06-0.10 с, поэтому при таком ритме QRS превышает 0.12 с.
- Отсутствует закономерность между комплексами QRS и зубцами P, потому что АВ-соединение не выпускает импульсы из желудочков, а предсердия могут возбуждаться из синусового узла, как и в норме.
- ЧСС менее 40 ударов в минуту.

Рисунок 1.12 Идиовентрикулярный ритм. Зубец P не связан с комплексом QRS.

d. Анализ проводимости.

Для достоверного учета проводимости обязательно учитывать скорость записи.

Для оценивания проводимости измеряют:

- Длительность зубца Р (отражает скорость проведения импульса по предсердиям), в норме до 0.1 с.
- Длительность интервала Р - Q (отражает скорость проведения импульса от предсердий до миокарда желудочков); интервал Р - Q = (зубец Р) + (сегмент Р - Q). В норме 0.12-0.2 с.
- Длительность комплекса QRS (отражает распространение возбуждения по желудочкам). В норме 0.06-0.1 с.
- Интервал внутреннего отклонения в отведениях V1 и V6, показанный на рис 1.13 – это время между началом комплекса QRS и зубцом R. В норме в V1 до 0.03 с и в V6 до 0.05 с. Используется в основном для распознавания блокад ножек пучка Гиса и для определения источника возбуждения в желудочках в случае желудочковой экстрасистолы (внеочередного сокращения сердца).

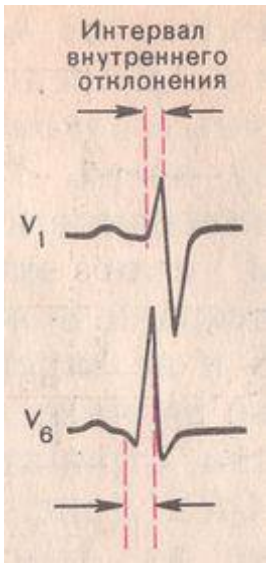

Рисунок 1.13 Измерение интервала внутреннего отклонения.

3) Установление электрической оси сердца.

В первой части цикла про ЭКГ объяснялось, что такое электрическая ось сердца и как ее определяют во фронтальной плоскости.

4) Оценка предсердного зубца Р.

В норме в отведениях I, II, avf, V2 - V6 зубец Р всегда положительный. В отведениях III, avl, V1 зубец Р может быть положительным или двухфазным

(часть зубца положительная, часть - отрицательная). В отведении avr зубец Р всегда отрицательный.

В норме длительность зубца Р не превышает 0.1 с, а его амплитуда - 1.5 - 2.5 мм.

Патологические отклонения зубца Р:

- Заостренные высокие зубцы Р нормальной продолжительности в отведениях II, III, avf характерны для гипертрофии правого предсердия, которая представлена на рисунке 1.14, например, при “легочном сердце”.
- Расщепленный с двумя вершинами, расширенный зубец Р в отведениях I, avl, V5, V6 характерен для гипертрофии левого предсердия, которая представлена на рисунке 1.15, как, например, при пороках митрального клапана.

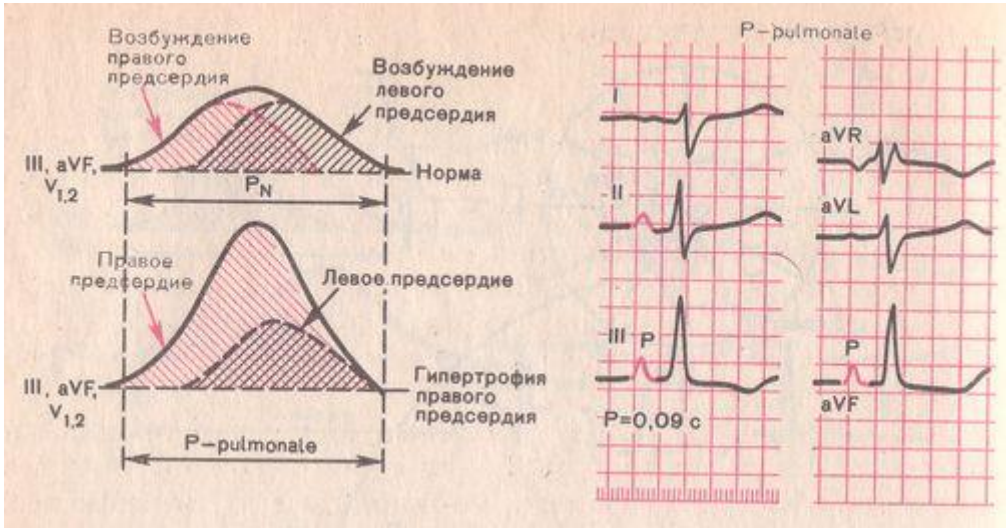

Рисунок 1.14 Формирование зубца Р (P-pulmonale) при гипертрофии правого предсердия.

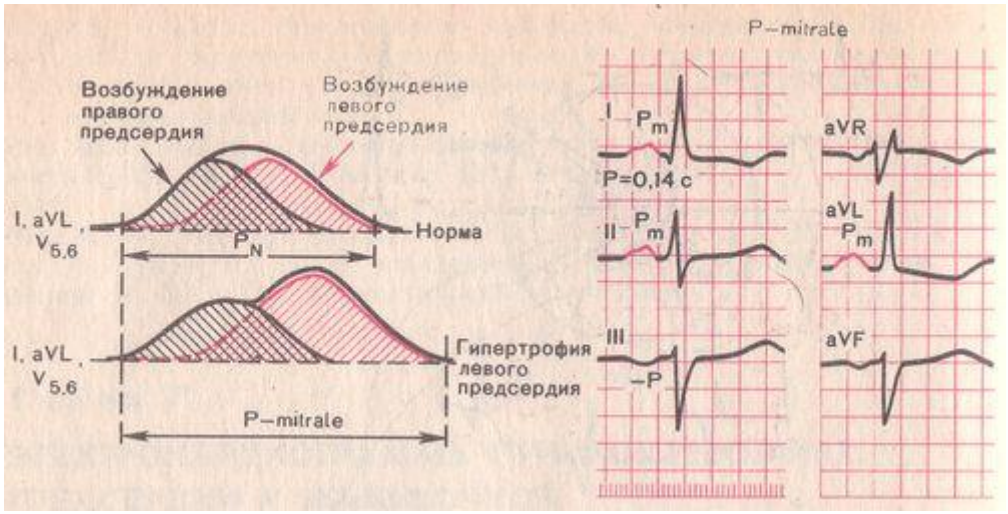

Рисунок 1.15 Формирование зубца P (P-mitrale) при гипертрофии левого предсердия.

Интервал P-Q: в норме 0.12-0.20 с.

Увеличение данного интервала бывает при нарушенном проведении импульсов через предсердно-желудочковый узел (атриовентрикулярная блокада, AV-блокада).

AV-блокада бывает 3 степеней:

- I степень - интервал P-Q увеличен, но каждому зубцу P соответствует свой комплекс QRS (выпадения комплексов нет).
- II степень - комплексы QRS частично выпадают, т.е. Не всем зубцам P соответствует свой комплекс QRS.
- III степень - полная блокада проведения в AV-узле. Предсердия и желудочки сокращаются в собственном ритме, независимо друг от друга. Т.е. Возникает идиовентрикулярный ритм.

5) Оценка желудочкового комплекса QRST:

а. Анализ комплекса QRS.

Максимальная длительность желудочкового комплекса равна 0.07-0.09 с (до 0.10 с). Длительность увеличивается при любых блокадах ножек пучка Гиса.

В норме зубец Q может регистрироваться во всех стандартных и усиленных отведениях от конечностей, а также в V4-V6. Амплитуда зубца Q в

норме не превышает 1/4 высоты зубца R, а длительность - 0.03 с. В отведении avr в норме бывает глубокий и широкий зубец Q и даже комплекс QS.

Зубец R, как и Q, может регистрироваться во всех стандартных и усиленных отведениях от конечностей. От V1 до V4 амплитуда нарастает (при этом зубец r_{v1} может отсутствовать), а затем снижается в V5 и V6.

Зубец S может быть самой разной амплитуды, но обычно не больше 20 мм. Зубец S снижается от V1 до V4, а в V5-V6 даже может отсутствовать. В отведении V3 (или между V2 - V4) обычно регистрируется “переходная зона” (равенство зубцов R и S).

b. Оценка сегмента RS - T

Сегмент S-T (RS-T) является отрезком от конца комплекса QRS до начала зубца T. Сегмент S-T особенно внимательно анализируют при ИБС, так как он отражает недостаток кислорода (ишемию) в миокарде.

В норме сегмент S-T находится в отведениях от конечностей на изолинии (± 0.5 мм). В отведениях V1-V3 возможно смещение сегмента S-T вверх (не более 2 мм), а в V4-V6 - вниз (не более 0.5 мм).

Точка перехода комплекса QRS в сегмент S-T называется точкой j (от слова junction - соединение). Степень отклонения точки j от изолинии используется, например, для диагностики ишемии миокарда.

c. Оценка зубца T.

Зубец T отражает процесс реполяризации миокарда желудочков. В большинстве отведений, где регистрируется высокий R, зубец T также положительный. В норме зубец T всегда положительный в I, II, avf, V2-V6, причем $T_I > T_{III}$, а $T_{V6} > T_{V1}$. В avr зубец T всегда отрицательный.

d. Оценка интервала Q - T.

Интервал Q-T называют электрической систолой желудочков, потому что в это время возбуждаются все отделы желудочков сердца. Иногда после зубца T регистрируется небольшой зубец U, который образуется из-за кратковременной повышенной возбудимости миокарда желудочков после их реполяризации.

6) Электрокардиографическое заключение.

Должно включать в себя:

1. Источник ритма (синусовый или нет).
2. Регулярность ритма (правильный или нет). Обычно синусовый ритм является правильным, хотя возможна дыхательная аритмия.
3. ЧСС.
4. Положение электрической оси сердца.
5. Наличие 4 синдромов:

- Нарушение ритма
- Нарушение проводимости
- Гипертрофия и/или перегрузка желудочков и предсердий
- Повреждение миокарда (ишемия, дистрофия, некрозы, рубцы)

Примеры заключений (не совсем полных, но реальных):

Синусовый ритм с ЧСС 65. Нормальное положение электрическое оси сердца. Патологии не выявлено.

Синусовая тахикардия с ЧСС 100. Единичная наджелудочная экстрасистолия.

Ритм синусовый с ЧСС 70 уд/мин. Неполная блокада правой ножки пучка Гиса. Умеренные метаболические изменения в миокарде.

1.2 Исследование автоматического непрерывного анализа электрокардиосигнала в приборах и системах кардиологического наблюдения

Программное обеспечение содержит в себе основополагающее значение при создании системы электрокардиографии высокого разрешения (ЭКГ ВР). Только алгоритмическое и программное обеспечение задает функциональные

способности диагностики и, в значительной степени, верность постановки диагноза в целом.

В таблице 1.1 представлен перечень и основные возможности программ для анализа и обработки сигнала ЭКГ от главных мировых компаний-производителей электрокардиографов.

Таблица 1.1

Интерпретации сигнала ЭКГ от компаний- производителей электрокардиографов.

Название программы	Производитель	Ключевые возможности
ECG interpretation software C [1]	Schiller (Швейцария)	Больше 100 разных вариантов интерпретации ЭКГ
Signal-Averaged ECG Software (SAECG) [2]	Schiller (Швейцария)	Анализ сигнал-усредненной ЭКГ
Heart Rate Variability (HRV) Software [3]	Schiller (Швейцария)	Анализ вариабельности сердечного ритма (ЭКГ)
Marquette 12SL [4]	GE Healthcare (США)	Анализ ЭКГ в 12 отведениях; уникальные критерии оценки ST- и QT-сегментов и зубца Т; автоматическое обнаружение аритмий
Cardiosoft [5]	GE Healthcare (США)	Расширенный анализ сегмента ST; анализ альтернансов зубца Т; автоматическое и ручное измерение интервалов ЭКГ; анализ аритмий по 2 отведениям; уникальные алгоритмы фильтрации шумов и выравнивания изолинии
FP-804 [6]	Fukuda (Япония)	Основные измерения: ЧСС, RR, PR, QRS, время QT, QTc, электрическая ось, SV, RV5(6); ввод информации о пациенте; 120 типов кодов интерпретации и 130 типов кодов Миннесоты

Следует выделить данные общие проблемы, которые свойственны многим диагностическим системам ЭКГ ВР [7]:

– трудность в ведении учета во процессе автоматического анализа ЭКГ сведений, которые сама ЭКГ не содержит, но обязательно (часто неявно) учитываемых врачом при интерпретации ЭКГ;

– диагностические ошибки. Имеются оценки, что в коммерческих системах автоматического анализа ЭКГ от 5 до 20 процентов автоматических заключений полностью или частично не совпадают с врачебными заключениями. В зависимости от реализации, система может быть склонна делать вполне определенные ошибки (гипер- или гиподиагностика определенной группы ЭКГ-заключений) или вообще не выявлять какую-либо электрокардиографическую патологию. Возможно, что улучшение диагностики одного класса нарушений может привести к ухудшению показателей в других классах[7];

– не эргономичный пользовательский интерфейс.

Отталкиваясь от проведенного анализа существующих программных инструментов автоматического анализа ЭКГ, можно выявить основные возможности, которыми должна обладать программа, обеспечивающая наиболее точный анализ и интерпретацию сигнала ЭКГ:

– внесение ЭКГ-сигнала и показ его на экране;

– диагностирование характерных структурных элементов ЭКГ;

– размещение отметок узловых точек сигнала;

– автоматизированное и ручное установление интервалов ЭКГ;

– фиксирование наиболее информативных параметров;

– оценка вариативности сердечного ритма;

– вывод результатов оценки в графическом виде;

– трактовка результатов оценки информативных параметров;

– автоматическое создание интерпретационной информации.

1.3 Обзорное исследование нынешнего уровня развития математических методов и алгоритмов автоматической обработки изображения графика ЭКГ

Исследованы имеющиеся методы решения проблемы нахождения QRS-комплекса ЭКГ, в фундаменте которых заложены последующие объединенные методы:

- расчет производных сигнала;
- линейная цифровая фильтрация;
- адаптивная цифровая фильтрация;
- согласованная фильтрация;
- вейвлет-преобразование;
- нейронные сети;
- синтаксические методы.

Зафиксировано, что, принимая во внимание особенности ЭКС при достаточно продолжительном кардиологическом мониторинге, преимущество должны иметь методы, основывающиеся на в наибольшей степени общих свойствах и характеристиках сигнала. Данному императиву в основе своей отвечают методы, базирующиеся на применении цифровой фильтрации [2].

Отмечены более обширно применяемые способы систематизации QRS-комплексов ЭКГ по типам их морфологий:

- на базе признаков формы, определяемых во временном отрезке;

Данные методы основываются на применении несложных логических правил по отношению к исходному ЭКС. Они характеризуются относительной простотой, но обладают невысокой устойчивостью к помехам и к изменчивости ЭКС, а также требуют дополнительной индивидуальной подстройки порога для каждого пациента для повышения надежности работы алгоритма.

- спектральные методы;

В данном подходе сигнал сначала пропускают через цифровой фильтр, частотная характеристика которого соответствует спектру QRS-комплекса, полученному путем усреднения спектральных оценок большого числа комплексов различной морфологии. На следующем этапе анализа предварительно обнаруживают QRS-комплекс и определяется его опорная точка.

- методы на базе корреляционных функций;

Указанные методы вычисляют в текущем периоде взаимную корреляционную функцию между входным ЭКС и одним или несколькими образцами желудочковых комплексов. Такой алгоритм обеспечивает хорошее качество обнаружения QRS-комплекса даже при наличии значительных помех, а также способны адаптироваться в процессе анализа к форме сигнала каждого конкретного пациента, но реализация данных методов требует

использования высокопроизводительного процессора и специализированных быстродействующих устройств.

- применение вейвлет-преобразования;
- применение разложения на главные компоненты;

Основаны на предварительной сегментации ЭКС алгоритмами сжатия сигнала, в ходе которой входной сигнал представляют последовательностью простейших элементов. Данные методы характеризуются наглядностью, удобством для программирования, универсальностью, но к недостаткам относится потеря части информации о сигнале при его сегментации, подверженность скачкообразному изменению результатов анализа при незначительных случайных отклонениях в форме входного сигнала.

- применение нейронных сетей.

Обозначены сильные и слабые стороны каждого из способов и предложено изучать достижимость решения вопроса систематизации формы QRS-комплекса модифицированным методом на базе признаков формы.

1.3 Типовая схема программы

Рассмотрим одну из типовых схем программ, которую используют для определения интервалов ЭКГ [41]. В рассматриваемой схеме (рис. 1.16) представлена общая схема программы, в которой можно выделить 7 блоков (модулей): блок моделирования ЭКГ сигнала (рис. 1.17), блок фильтрации сигнала (рис. 1.18), блок измерения R-R интервала (рис. 1.19), блок нахождения зубца P (рис. 1.20), блок измерения P-Q интервала (рис. 1.21)**,** блок измерения QRS комплекса (рис. 1.22), блок измерения Q-T интервала и S-T сегмента (рис. 1.23). Разбиение на модули обосновываются необходимостью, поскольку программа получается очень большой и проводить ее отладку затруднительно.

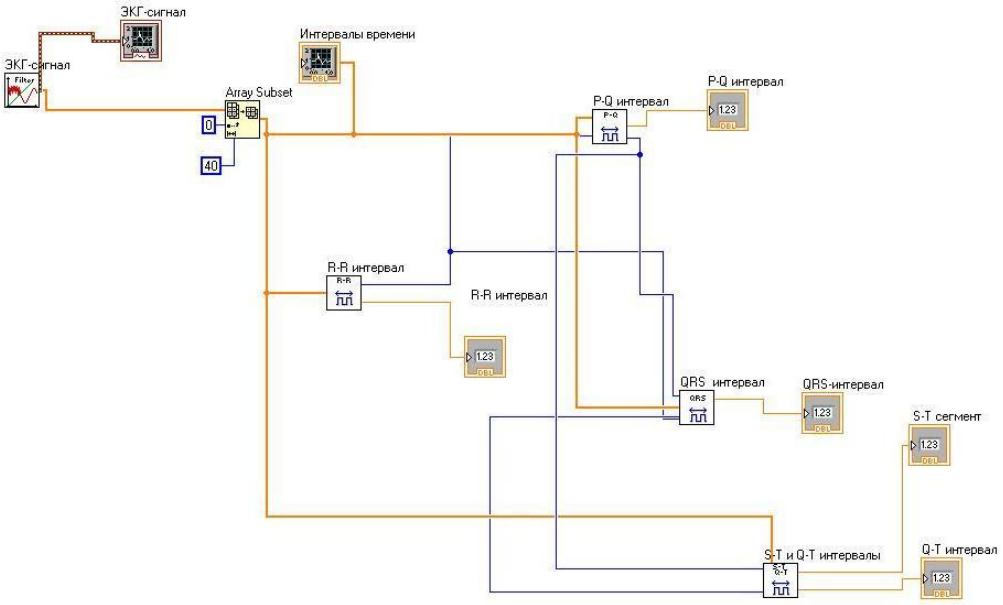

Рисунок 1.16 Общая схема программы

Для моделирования ЭКГ сигнала в рассматриваемой программе осуществлялась генерация сигнала с помощью цикла For Loop с числом итерации 150 (была использована математическая модель сигнала, описанная М.В. Абрамовым в статье «Аппроксимация экспонентами временного кардиологического ряда на основе ЭКГ» [41]).

В блоке фильтрации можно видеть, что последовательно используются фильтр высокой частоты, полосовой фильтр и фильтр низкой частоты.

Согласно программе идентификация кардиологических комплексов начинается с получения оценки пульса pulse как расстояния между максимумами амплитуды зубца R. Положение двух последних найденных зубцов R запоминается. Как только расстояние между текущим моментом времени и t_{R1} превысит RRmin, по величине данной разности производится оценка пульса.

Учитывая строгую хронологическую последовательность зубцов, первыми определяются параметры зубца P. Параметры зубцов S и R определяются только после того, как найдены параметры зубца P, а параметры

зубцов Q и T –только после того, как найдены параметры зубца R.

После нахождения P-зубца для измерения P-Q интервала применяется схема, показанная на рис. 1.21.

Анализ данной программы позволил выявить ряд недостатков, как связанных с программным средством, в котором была реализована программа, так и с возможностью ее использования вне медицинских центров.

Для использования в быстрой предварительной обработке ЭКГ, а также в личных целях (домашних условиях) предложена программа, реализованная в среде MS Visual C++.

1.3.1 Блок моделирование ЭКГ сигнала

Для моделирования ЭКГ сигнала была использована математическая модель сигнала, описанная М.В. Абрамовым в статье «Аппроксимация экспонентами временного кардиологического ряда на основе ЭКГ» [41]. Выглядит она следующим образом:

$$F_{\Sigma} = -0.11 \cdot e^{-(x-58)^2 \cdot \begin{vmatrix} 0.037, x \le 58 \\ 0.01, x > 58 \end{vmatrix}} + e^{-(x-74)^2 \cdot \begin{vmatrix} 0.03, x \le 74 \\ 0.05, x > 74 \end{vmatrix}} - 0.19 \cdot e^{-(x-89)^2 \cdot \begin{vmatrix} 0.04, x \le 89 \\ 0.01, x > 89 \end{vmatrix}} - 0.12 \cdot e^{-(x-110)^2 \cdot \begin{vmatrix} 0.006, x \le 110 \\ 0.0005, x > 110 \end{vmatrix}} + \\ + 0.22 \cdot e^{-(x-195)^2 \cdot \begin{vmatrix} 0.0015, x \le 195 \\ 0.0033, x > 195 \end{vmatrix}}$$

Генерация сигнала производится с помощью цикла For Loop с числом итерации 150. На рис. 1.17 можно увидеть часть созданной программы.

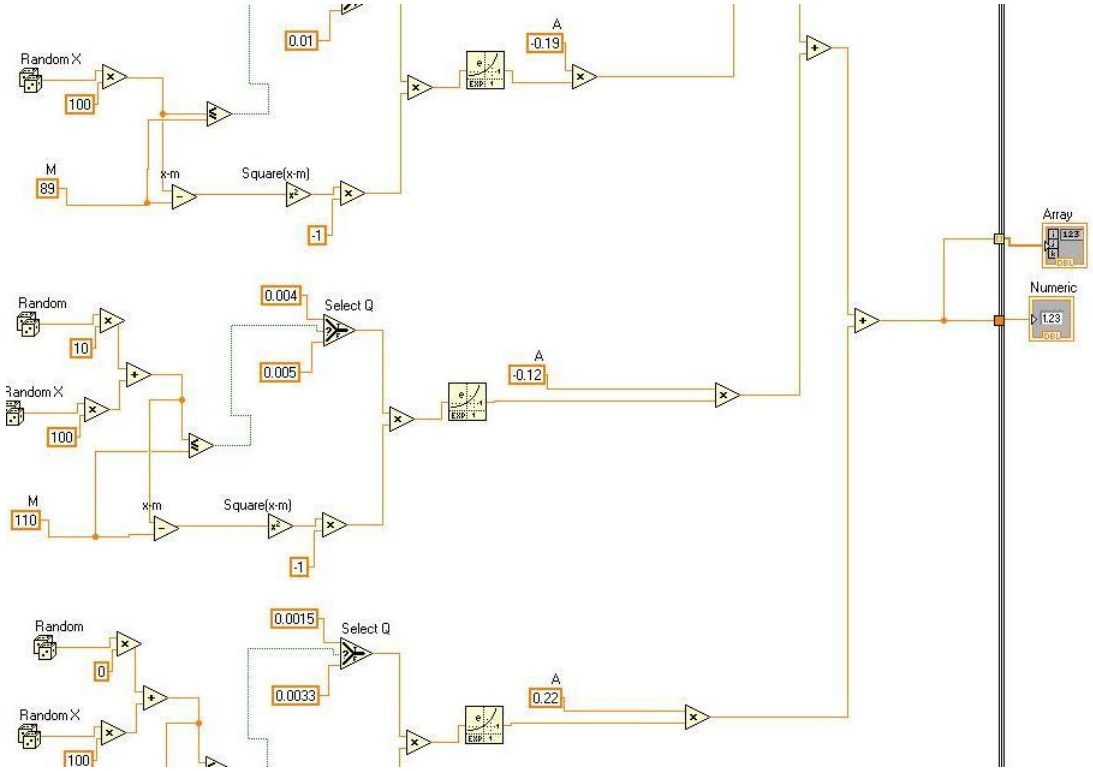

Рисунок 1.17. Блок моделирование ЭКГ сигнала

Блок фильтрации сигнала

На рис. 1.18 представлена схема блока фильтрации сигнала.

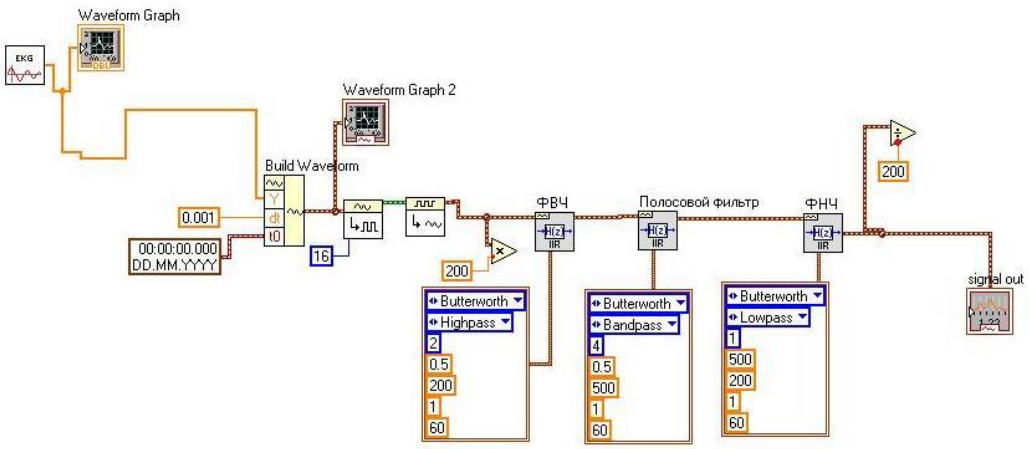

Рисунок 1.18. Блок фильтрации сигнала

Блок измерения R-R интервала

На рис. 1.19 представлена схема блока измерения R-R интервала.

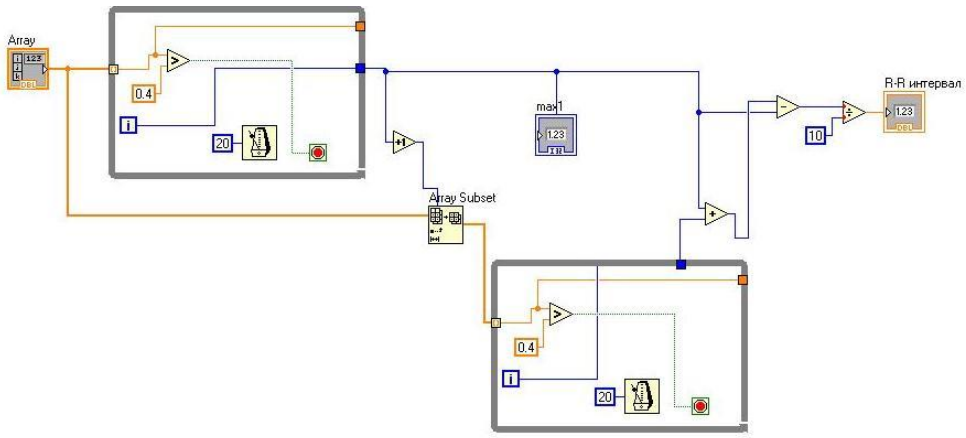

Рисунок 1.19. Блок измерения R-R интервала

Идентификация кардиологических комплексов начинается с получения оценки пульса pulse как расстояния между максимумами амплитуды зубца R. Положение двух последних найденных зубцов R запоминается. Как только расстояние между текущим моментом времени и t_{R1} превысит RRmin, по величине данной разности производится оценка пульса.

Блок измерения P-Q интервала

На рис. 1.20 представлена схема нахождения P зубца.

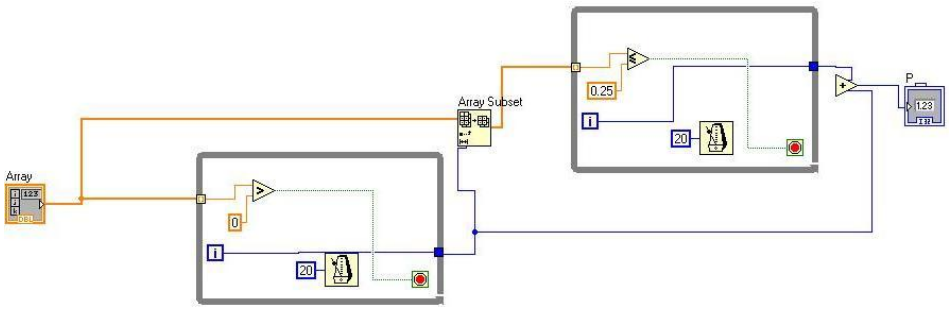

Рисунок 1.20. Нахождение P зубца

Учитывая строгую хронологическую последовательность зубцов, первыми определяются параметры зубца P. Параметры зубцов S и R определяются только после того, как найдены параметры зубца P, а параметры зубцов Q и T –только после того, как найдены параметры зубца R.

После нахождения P-зубца для измерения P-Q интервала применяется схема, показанная на рис. 1.21.

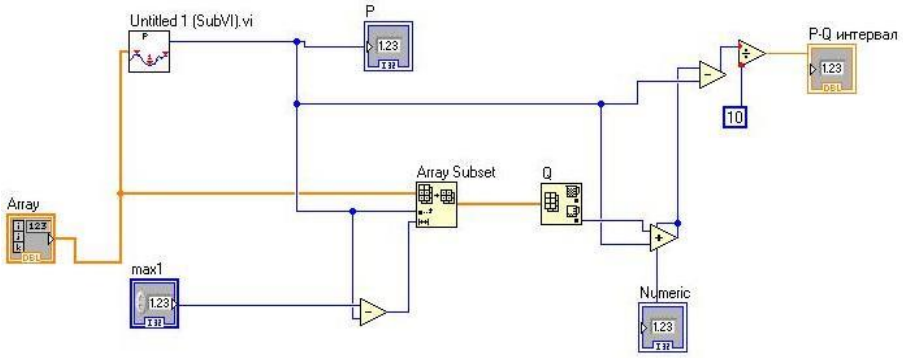

Рисунок 1.21. Блок измерения P-Q интервал

Блок измерения QRS комплекса

На рис. 1.22. показан блок измерения QRS комплекса.

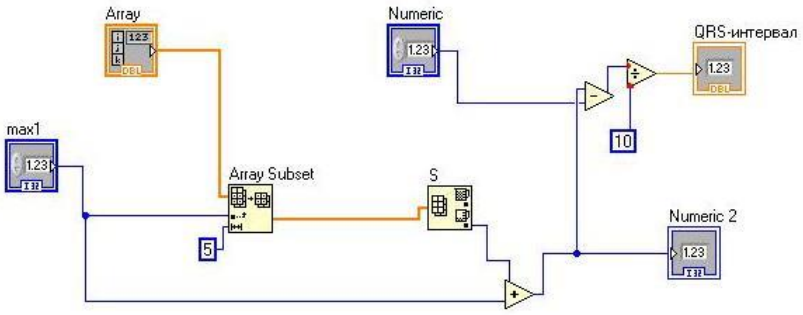

Рисунок 1.22. Блок измерения QRS комплекса

Блок измерения Q-T интервала и S-T сегмента

На рис. 1.23 представлен блок измерения Q-T интервала и S-T сегмента.

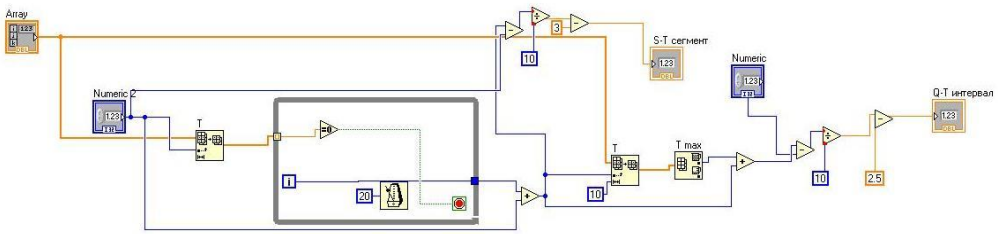

Рисунок 1.23. Блок измерения Q-T интервала и S-T сегмента

Проведенное в данном пункте моделирование позволило выявить недостатки в предложенной схеме и сформулировать требования к разработке на C++.

2. РАЗРАБОТКА АЛГОРИТМОВ, ПРОЦЕДУР И МЕТОДОВ АНАЛИЗА ИЗОБРАЖЕНИЙ ЭКГ

2.1 Обзор программ для анализа и интерпретации сигнала ЭКГ

Программное обеспечение имеет основополагающее значение при построении системы электрокардиографии высокого разрешения (ЭКГ ВР). Именно алгоритмическое и программное обеспечение определяют функциональные возможности диагностики и в значительной степени правильность постановки диагноза в целом.

Можно выделить следующие общие проблемы, присущие многим диагностическим системам ЭКГ ВР [7]:

– сложность учета во время автоматического анализа ЭКГ сведений, не содержащихся в самой ЭКГ, но обязательно (часто неявно) учитываемых врачом при интерпретацииЭКГ;

– диагностические ошибки. Имеются оценки, что в коммерческих системах автоматического анализа ЭКГ от 5 до 20 процентов автоматических заключений полностью или частично не совпадают с врачебными заключениями. В зависимости от реализации, система может быть склонна делать вполне определенные ошибки (гипер- или гиподиагностика определенной группы ЭКГ заключений) или вообще не выявлять какую-либо электрокардиографическую патологию. Вероятно, что улучшение диагностики одного класса нарушений может привести к ухудшению показателей в других классах;

– неудобный пользовательский интерфейс.

Исходя из проведенного анализа существующих программных средств автоматического анализа ЭКГ, выявлены основные возможности, которыми должна обладать программа, обеспечивающая анализ и интерпретацию сигнала ЭКГ:

– загрузка ЭКГ-сигнала и отображение его на экране;

– распознавание характерных элементов ЭКГ;

– расстановка маркеров узловых точек сигнала;

– автоматическое и ручное измерение интервалов ЭКГ;

– измерение информативных параметров;

- расчет ЧСС.

2.2 Общий алгоритм работы программы

Предлагается алгоритм, состоящий из следующих модулей (рис. 2.1):

- загрузка изображения (кардиограммы), в данном модуле предполагается считывание кардиограммы;

- фильтрация изображения (удаление шумов), в случае некачественного изображения в данном модуле производится удаление шумов;

- перевод в градации серого.

Рисунок 2.1 Обобщенная схема предлагаемого алгоритма

Перевод изображения в оттенки серого производится следующим образом:

1. Разложение значений цвета каждого пикселя на изображении согласно цветовой схеме RGB (red , green , blue) на значения трех цветовых составляющих: R (красного), G (зеленого), B (синего).

2. Выравнивание значений составляющих цветовой схемы.

Для выравнивания значений цветовых составляющих может применяться следующий подход - значения цветовых составляющих усредняются или приравниваются к яркости, рассчитываемой по формуле:

$$y_i = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B,$$

где yi – значение яркости пикселя, R , G , B – значения соответствующих составляющих цвета пикселя.

В таком случае получаемое в оттенках серого изображение обладает меньшей контрастностью. Для выравнивания значений предлагается все три цветовых составляющих приравнивать к наибольшему из значений цветовой схемы. Так, если наибольшее значение у конкретного пикселя имеет красное составляющее, то значения синего и зеленого составляющих приравнивается к красному.

Итогом данного этапа является цифровое изображение в оттенках серого (рис. 2.2).

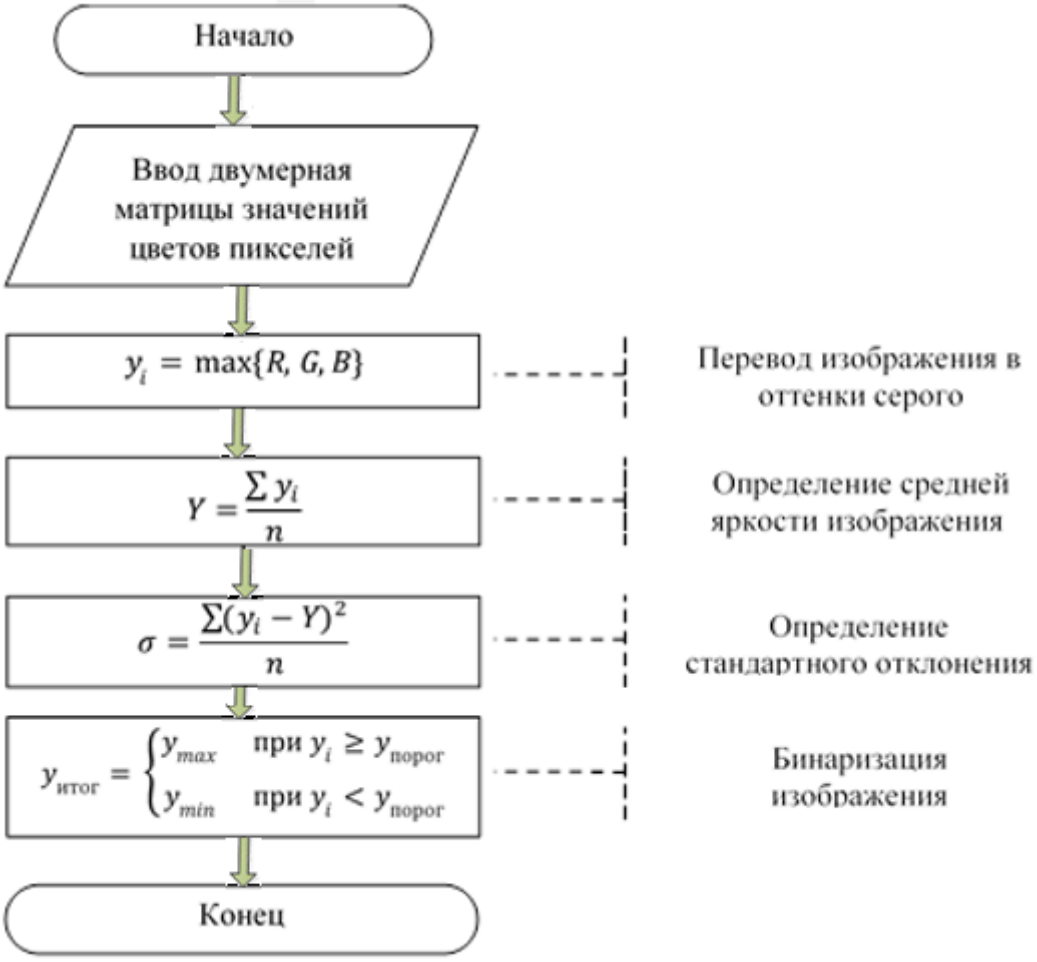

Рисунок 2.2 Схема алгоритма перевода в оттенки серого

- сегментация изображений

- поиск зубцов R, который происходит следующим образом: в выделенной области происходит поиск локальных максимумов, при этом перебираются все точки выделенной области (рис. 2.3)

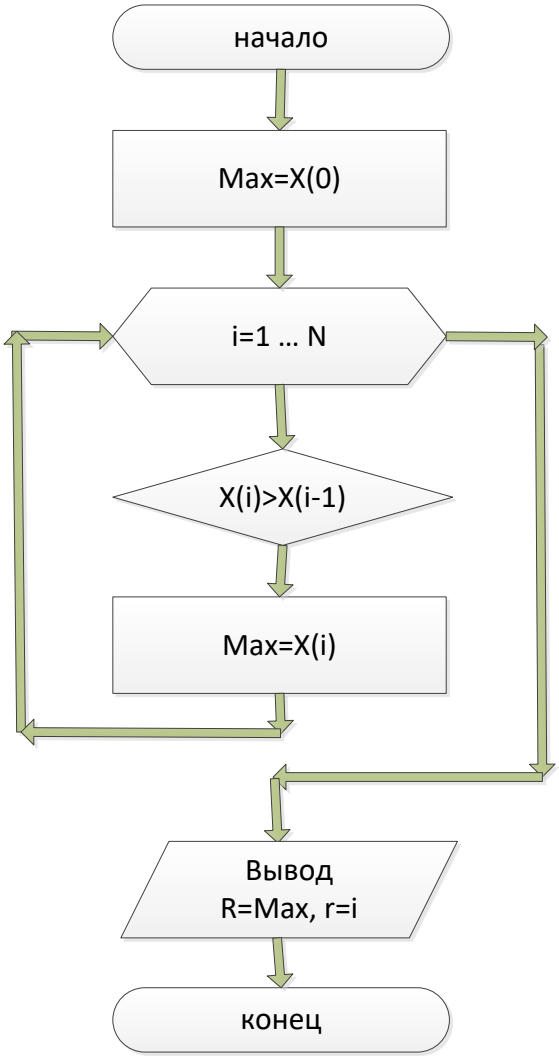

Рисунок 2.3 Поиск зубца R

- поиск зубцов Q, который происходит следующим образом: в выделенной области происходит поиск первого локального минимума слева от R (рис. 2.4);

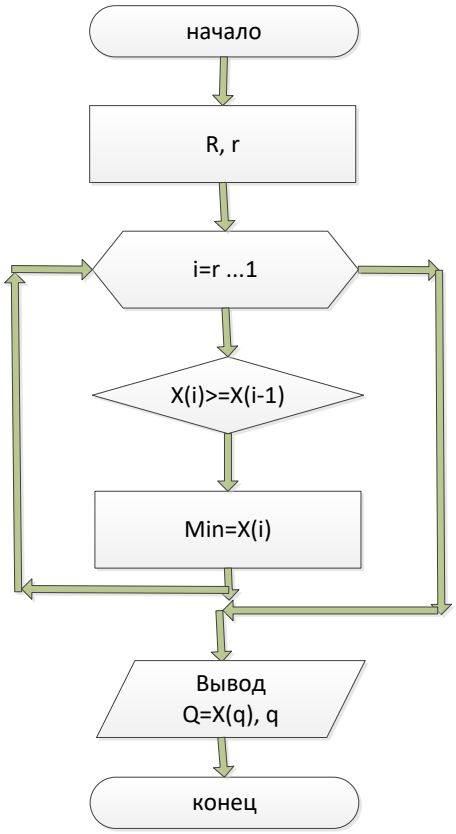

Рисунок 2.4 - поиск зубца Q

- поиск зубцов S, который происходит следующим образом: в выделенной области происходит поиск первого локального минимума справа от R (рис. 2.5);

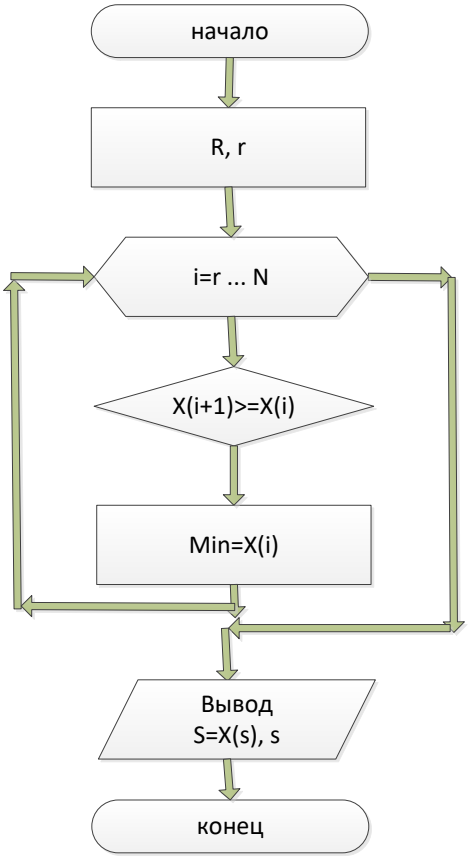

Рисунок 2.5 - поиск зубца S

- поиск зубцов T, который происходит следующим образом: в выделенной области происходит поиск первого локального максимума справа от S (рис. 2.6);

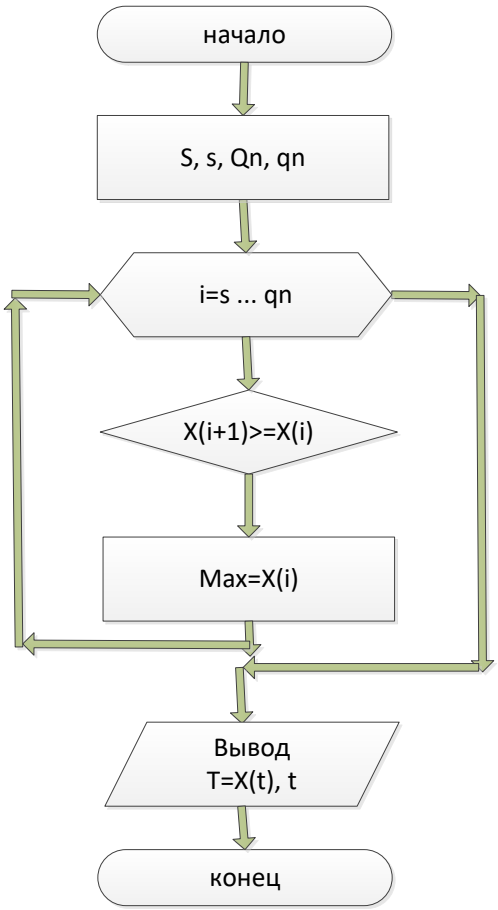

Рисунок 2.6 - поиск зубца T

- поиск зубцов P, который происходит следующим образом: в выделенной области происходит поиск первого локального максимума слева от Q (рис. 2.7);

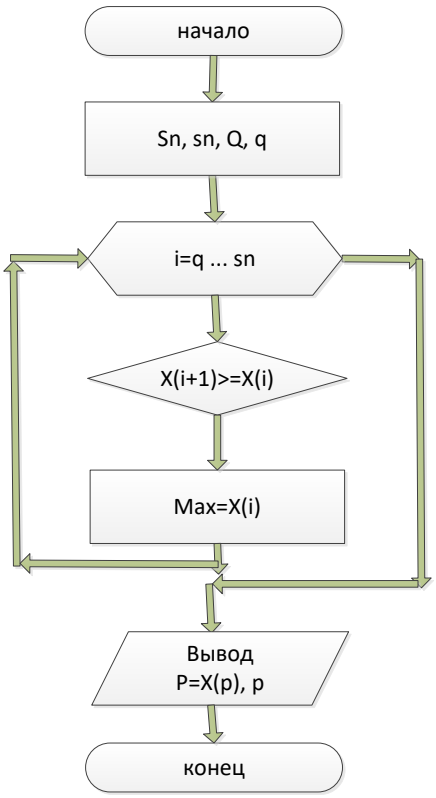

Рисунок 2.7 – поиск зубца P

2.3 Исследование алгоритмов обнаружения QRS-комплекса ЭКГ

Известен способ выделения R-R интервалов [24], заключающийся в выполнении следующей последовательности действий. ЭКС фильтруют, дифференцируют, осуществляют двухполупериодное выпрямление для исключения пропуска QRS-комплекса при отрицательном зубце R, детектируют с помощью пикового детектора, из детектированного сигнала формируют порог сравнения, с которым сравнивают ЭКС, момент сравнения ЭКС с пороговым уровнем принимают за начало очередного R-R интервала (кардиоцикла).

Данный способ имеет следующие недостатки:

1) в ряде случаев амплитуда зубца R QRS-комплекса может быть сравнима с амплитудой зубца Т (это выявлено в первом стандартном отведении даже у пациентов с нормальной электрокардиограммой), что затрудняет надежное выделение QRS-комплекса;

2) в кардиограммах с расщепленным зубцом R надежность выделения начала кардиоцикла снижается.

Наиболее близким к предлагаемому способу (прототипом) является способ определения начала кардиоцикла. реализованный в устройстве [15], заключающийся в том, что ЭКС дифференцируют, чтобы обострить QRS-комплекс, формируют из полученного сигнала пороговый уровень и сравнивают продифференцированный сигнал первый раз с пороговым уровнем и второй раз с результатом первого сравнения. По результатам сравнений формируют "временное окно", позволяющее исключить ложные выделения QRS-комплекса.

Данному способу присущи следующие недостатки:

1) надежное выделение QRS-комплекса может быть достигнуто только при надежном выделении первого (стартового) QRS-комплекса;

2) действие на ЭКС низкочастотных аддитивных помех (поляризация электродов, дыхательные волны, артефакты и т.п.) снижает надежность выделения начала кардиоцикла; попытка убрать низкочастотную аддитивную помеху с помощью фильтра высоких частот не даст результата, так как частота среза такого фильтра должна иметь малое значение (0,05 Гц в соответствии со стандартами на электрокардиографы), и действие артефактов приводит к возникновению переходного процесса, который длится тем дольше, чем ниже частота среза фильтра, во время переходного процесса надежность выделения QRS-комплекса снижается.

Предлагаемый способ позволяет устранить указанные недостатки и обеспечить надежное выделение начала очередного кардиоцикла.

Анализ электрокардиограмм с различными отклонениями от нормы показал, что при снятии электрокардиограммы в условиях поликлиники или стационара, то есть при спокойном состоянии пациента, наиболее стабильным участком электрокардиосигнала является часть изолинии между зубцами T и P. На ЭКС можно выделить еще две области, лежащие на изолинии: сегмент PQ - отрезок от окончания зубца P до начала зубца Q, сегмент ST - отрезок от конца комплекса QRS до начала зубца T. Сравнение длительностей указанных отрезков показывает, что в спокойном состоянии пациента длительность отрезка TP существенно превышает длительность сегмента PQ и сегмента ST. Указанные соотношения сохраняются и при действии на ЭКС аддитивной низкочастотной помехи. Это обстоятельство и положено в основу предлагаемого способа выделения начала кардиоцикла.

Суть предлагаемого способа заключается в следующем. Очищенный от высокочастотных помех и помехи промышленной частоты электрокардиосигнал дифференцируют два раза и сравнивают полученный сигнал с двумя пороговыми уровнями. Один уровень устанавливают на Δвыше нулевой линии, а другой - на Δниже нулевой линии. Учитывая временные и амплитудные соотношения между различными частями электрокардиосигнала и выбирая соответствующие постоянные дифференцирования, значения уровней можно установить равными амплитуде зубца P. Если амплитуда дважды дифференцированного сигнала оказывается между пороговыми уровнями, начинают счет тактовых импульсов, частоту повторения которых можно выбрать равной общепринятой в электрокардиографии частоте дискретизации 500 имп/с. При достижении в результате счета заданного числа N принимают за начало очередного кардиоцикла положение на оси времени последнего из сосчитанных, то есть N-го импульса. Если же амплитуда

дважды дифференцированного ЭКС выйдет за пороговые уровни раньше, чем при счете достигнуто число N, то счет начинают заново.

Число N выбирают таким, что значения N при счете можно было бы достигнуть только на отрезке ЭКС между зубцами T и P.

Предложенный способ позволяет надежно выделить начало каждого кардиоцикла независимо от возможных отклонений от нормы параметров (формы, амплитуды, длительности) зубцов кардиосигнала, в частности QRS-комплекса, и дрейфе изолинии, обусловленном действием на ЭКС аддитивных низкочастотных помех (влияние дыхания, артефакты, временной дрейф и т.п.), что способствует улучшению условий последующей обработки кардиосигнала (вычисление временных параметров отдельных элементов кардиосигнала, вычисление длительности кардиоциклов и т.п.).

2.4 Алгоритм вычисления частоты сердечных сокращений

При расчете частоты сердечных нарушений (ЧСС) может быть несколько подходов:

- брать в расчет все выделенные кардиоциклы,
- брать в расчет только кардиоциклы с QRS-комплексами из класса "норма",
- брать в расчет только кардиоциклы нормальные по длительности (отличие от соседних не более 20%),
- брать в расчет только кардиоциклы нормальные по длительности (отличие от соседних не более 20%) и с QRS-комплексами из класса "норма".

При первом подходе есть вероятность увеличения значения ЧСС за счет ложных срабатываний алгоритма распознавания QRS-комплексов (деление одного кардиоцикла на два, даже при нормальном ритме).

Третий и четвертый подходы являются слишком строгими и в них происходит занижение ЧСС либо невозможность его расчета при сложных аритмиях.

При втором подходе считаются длительности только между нормальными по форме QRS-комплексами, а соотношение длительностей кардиоциклов может быть произвольным. Второй вариант предпочтительней, т. к. информация о ритме не искажается.

При расчете ЧСС производится усреднение длительности между QRS комплексами класса "норма" по форме (остальные в расчет не берутся) на основе нескольких QRS-комплексов (например, от 2 до 15) поступивших по времени последними. Для сохранения актуальности значения ЧСС следует брать QRS-комплексы пришедшие не позднее какого-то времени (например, 30 секунд). Затем значение усредненной длительности переводится в значение количества комплексов за минуту. Таким образом, при определении ЧСС будем использовать второй подход, при этом выделять будем 2-5 QRS-комплекса).

Блок-схема алгоритма показана на рис. 2.8. Алгоритм содержит в себе две частей: совокупность статистических данных в блоке 1 и оценка параметров ЭКГ в блоках 2 и 3 [9].

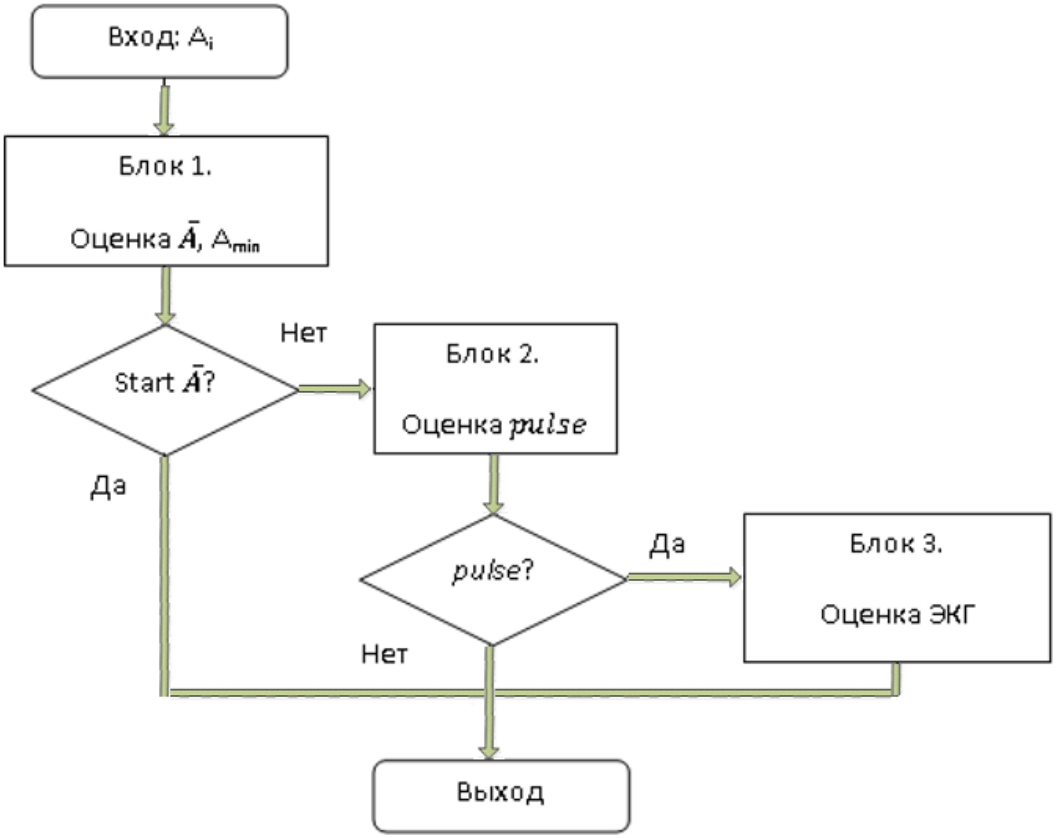

Рисунок 2.8 Блок-схема итеративного алгоритма идентификации кардиографических комплексов

Работа алгоритма оценки параметров ЭКГ проиллюстрирована на схеме на рис. 2.9.

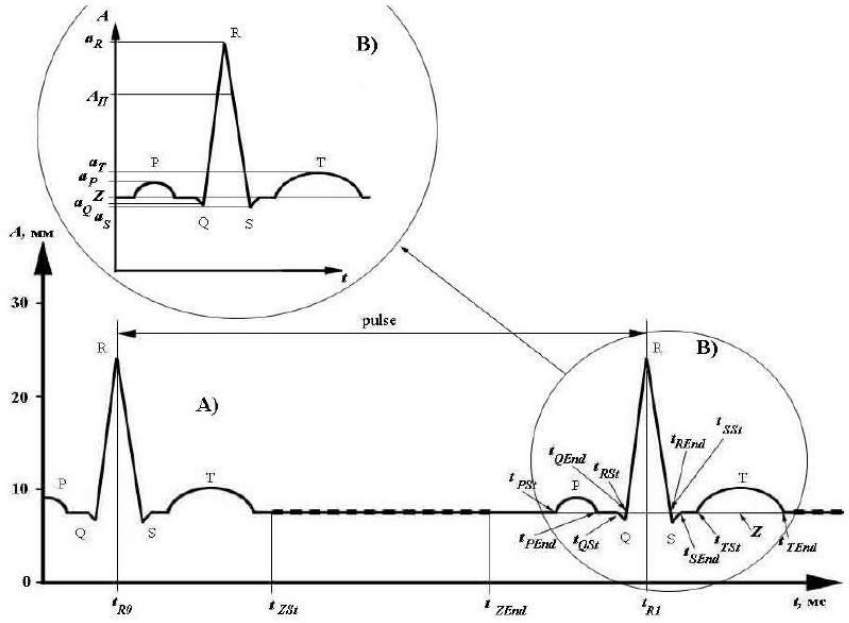

Рисунок 2.9 Схема обозначений оцениваемых параметров ЭКГ

Для определенности ниже будет рассмотрен случай, который представлен на рис. 2.9, при котором зубец R направлен вверх. Настоящее направление зубца R выявляется по величине асимметрии A_{min}, и A_{max} относительно среднего значения $\bar{A}$.

Определение кардиологических комплексов стартует с получения программой оценки пульса pulse как расстояния между максимумами амплитуд зубца R. Локация t_R зубца R определяется на основании из следующих двух условий. Первое условие: наличие местного максимума амплитуды ЭКГ, который удовлетворяющет условию $a_R > A_П$, где $A_П = A_{max} – (\bar{A} – A_{min})$. Второе условие: амплитуда других местных максимумов на интервале времени протяженностью от t_R до t_R + RRmin меньше a_R, где RRmin – минимальный периоду ЭКГ, соответствующий частоте сердечного ритма, которая равняется 200 ударов в минуту.

Местоположение двух из последних зубцов R – текущего t_{R1} и предыдущего t_{R0} — запоминаются. В момент, когда расстояние между настоящим моментом времени и t_{R1} превышает RRmin, по величине разности ($t_{R1} – t_{R0}$) осуществляется оценка пульса. В случае, если ($t_{R1} – t_{R0}$) превышает RRmax, то оценивание пульса не выполняется, и все расчеты в блоках 2 и 3 начинаются заново.

Сразу после того, как оценка пульса получена (условие <*pulse ?*> выполняется, см. рис. 2.8) в блоке 3 начинается определение следующих параметров ЭКГ (см. рис. 2.9):

Z – уровень изоэлектрической линии;

a_P, [t_{PSt}, t_{PEnd}] – амплитуда и расположение на временной шкале зубца P;

a_Q, [t_{QSt}, t_{QEnd}] – амплитуда и расположение на временной шкале зубца Q;

a_R, [t_{RSt}, t_{REnd}] – амплитуда и расположение на временной шкале зубца R;

a_S, [t_{SSt}, t_{SEnd}] – амплитуда и расположение на временной шкале зубца S;

a_T, [t_{TSt}, t_{TEnd}] – амплитуда и расположение на временной шкале зубца T.

Все вышеперечисленные параметры ЭКГ должны быть определены строго в том же порядке, как они указаны выше.

Показатель изоэлектрической прямой Z должен равняться среднему числу амплитуды ЭКГ на временном интервале [t_{ZSt}, t_{ZEnd}], который выбран для оценивания уровня изоэлектрической прямой. Момент времени t_{ZSt} избирается таким путем, чтобы он располагался сразу после зубца T текущего комплекса, а момент t_{ZEnd} – должен предшествовать зубцуP следующего комплекса PQRST:

$$t_{ZSt} = t_{TEnd} + L_{QRS0.5,X,}$$

$$t_{ZEnd} = t_R + \text{pulse} – (L_{RT} + L_{QRS0.5,X}),$$

где $t_R = t_{R0}$ – время, соответствующее максимуму текущего зубца R, полученное в блоке 2;

$L_{QRS0.5,X}$ – половина максимально возможной длительности комплекса QRS, равная 0,1 с;

$L_{RT} = (t_T - t_R)$ – расстояние между максимумами зубцов R и T;

t_T – время, соответствующее максимуму текущего зубца T.

В случае начала оценок в блоке 3, время t_T текущего зубца T определяется на интервале $[t_R+L_{QRS0.5,X}, t_R + L_{QRST,X}]$ и соответствует максимальному локальному максимуму на этом интервале, а время t_{TEnd} – окончания зубца T, полагается равным $t_{TEnd} = t_R+L_{QRST,X}$, где $L_{QRST,X}$ – максимально возможная длительность комплекса QRST, равная 0,42 с.

После того, как получена оценка уровня изоэлектрической линии Z, начинается поиск и оценка параметров зубцов P, Q, R, S и T.

Предполагается, что в кардиограмме зубцы P, R и T всегда присутствуют, а S и Q – могут отсутствовать.

Для зубцов, направленных вверх, находятся интервалы времени, для которых амплитуда ЭКГ выше изоэлектрической линии ($A_i > Z$), а для зубцов направленных вниз – ниже изоэлектрической линии ($A_i < Z$). Одновременно определяются амплитуда и положение вершины зубца, равные амплитуде и положению соответствующего экстремума.

Учитывая строгую хронологическую последовательность зубцов, первыми определяются параметры зубца P. Параметры зубцов S и R определяются только после того, как найдены параметры зубца P, а параметры зубцов Q и T –только после того, как найдены параметры зубца R. Если при этом амплитуда зубца R меньше порога $A_П$, то обнаруженный участок ЭКГ относится к зубцу P, а поиск зубцов Q и R возобновляется сначала.

Полученные ранее параметры зубца P могут быть заменены параметрами отбракованного зубца R, если его амплитуда меньше, чем амплитуда отбракованного зубца R.

Для полной идентификации зубцов P, Q, S и T используется привязка их времени существования к периоду сердечного ритма, а именно, оценка параметров этих зубцов производится в интервалах времени, привязанных к положению на временной оси зубцов R:

$[(t_R + pulse - L_{QRS0.5,X} - L_{PQ,X}), (t_R + pulse)]$ –интервал поиска зубца P;

$[(t_R + pulse - L_{QRS0.5,X}), (t_R + pulse)]$ – интервал поиска зубца Q;

$[t_R, t_R + L_{QRS0.5,X}]$ – интервал поиска зубца S;

$[t_R, t_R + L_{QRST,X}]$ – интервал поиска зубца T.

В приведенных выше выражениях t_R –положение на оси времени зубца R предыдущего (зубцы P и Q), или текущего (зубцы S и T) комплексов, а $L_{PQ,X}$ – максимально возможная длительность комплекса PQ, равная 0,2 с.

Отметим, что для полной идентификации зубца R в блоке 3 не задается явно временной интервал поиска. Как это было описано выше, его положение на оси времени ищется с применением процедуры отбраковки по амплитуде только после того, как найден зубец P. Длительности комплексов PQ, QRS и QRST полагаются равными разности времен окончания и начала крайних зубцов, составляющих комплекс. [9]

3. Результаты работы

3.1 Реализация информационной системы

Итогом разработки стало приложение, реализованное в среде разработки MS Visual Studio 2015 на языке программирования C++, которое производит подсчет частоты сердечных сокращений, которое функционирует следующим образом.

При открытии приложения появляется окно ввода основных данных, которое показано на рисунке 3.1.

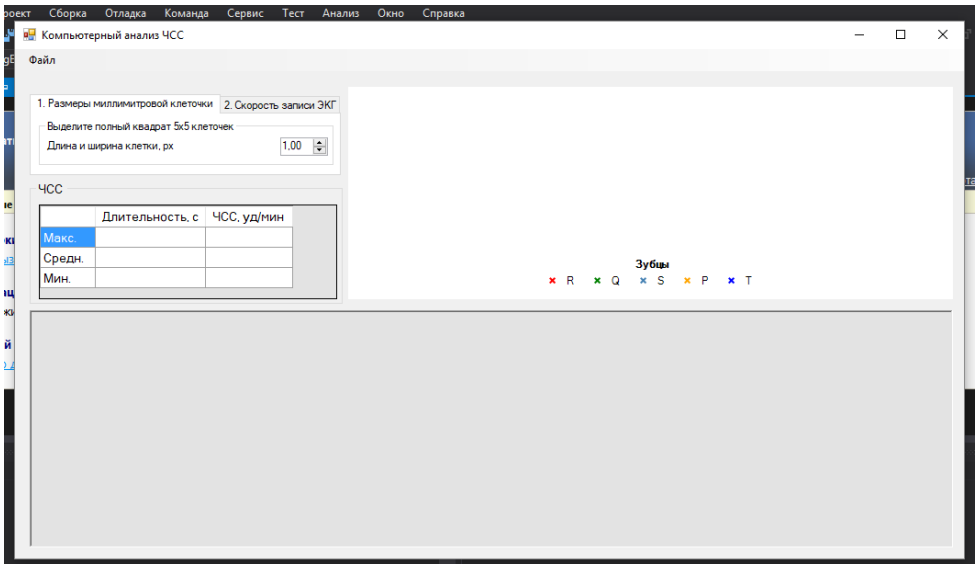

Рисунок 3.1. Окно ввода входных данных.

Данное окно является основным в графическом интерфейсе программы и имеет две вкладки «Размеры миллиметровой клеточки» и «Скорость записи ЭКГ» (рис. 3.2).

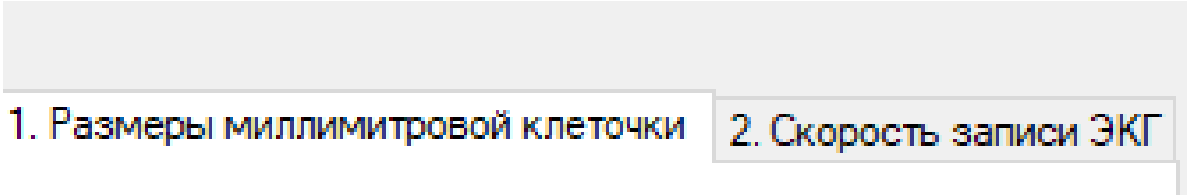

Рисунок 3.2. Основные функциональные вкладки интерфейса.

Для того, чтобы запустить алгоритм работы, необходимо загрузить изображение ЭКГ, которое должно иметь расширение .jpeg. Это можно сделать с помощью пунктов меню «Файл» - «открыть» и выбрать необходимое изображение ЭКГ. В итоге окно программы примет следующий вид (рис. 3.3).

Рисунок 3.3. Окно входных данных с загруженной ЭКГ-лентой.

Далее, для того, чтобы определить размерность ЭКГ-ленты в пикселях необходимо выделить область 5 на 5 клеток, которая будет использована в дальнейшем алгоритме обработки. Если будет выделена область меньше заданной, то появится сообщение об ошибке (рис. 3.4).

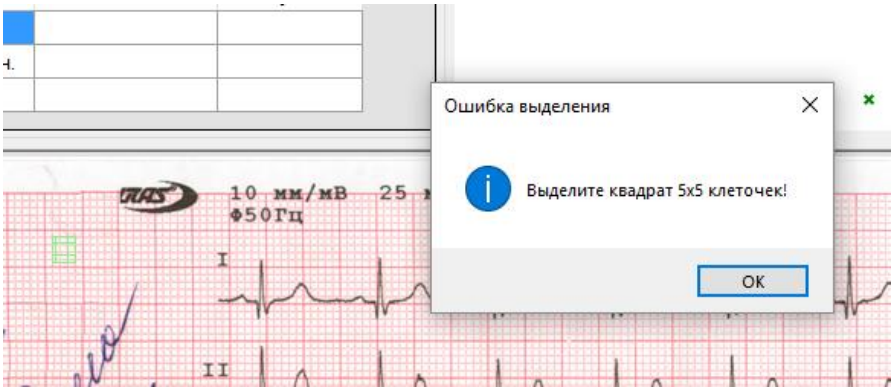

Рисунок 3.4. Сообщение об ошибке.

После того, как тестовая область была выделена, в блоке «Длина и ширина клетки. px» появляется значение, определяющее длину и ширину клетки на ленте ЭКГ (рис. 3.5) и можно переходить к следующей вкладке для выделения области для определения ЧСС.

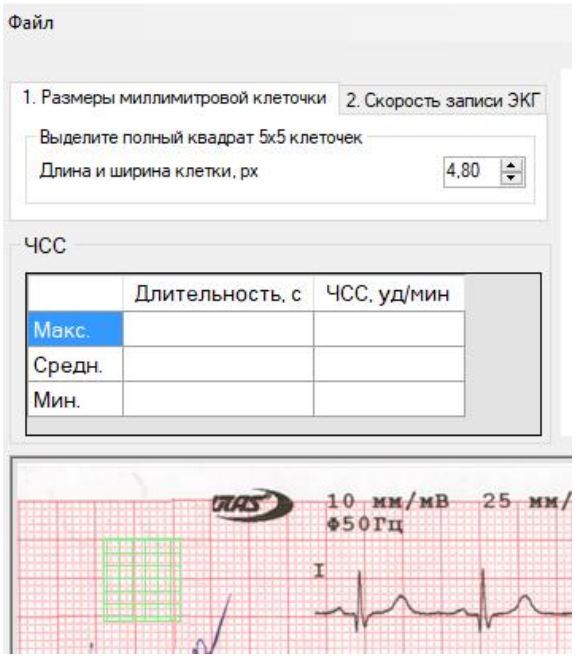

Рисунок 3.5. Результат записи начальных значений.

Для получения результата работы следует перейти на вкладку «Скорость записи ЭКГ». На ней нужно внести скорость записи ЭКГ, которая всегда отображается при записи непосредственно кардиограммы. На приведенном

примере можно увидеть, что скорость записи ЭКГ равняется 25 мм/с. Поэтому, это же значение вносится в блок «Скорость мм/с» (рис. 3.6).

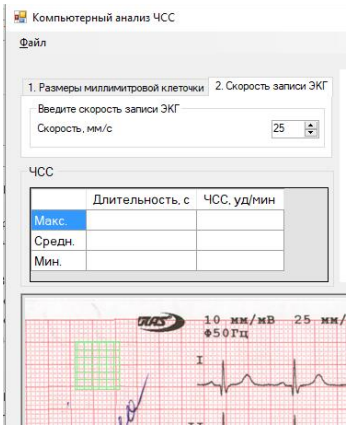

Рисунок 3.6. Работа на вкладке «Скорость записи ЭКГ»

Обработка кардиограммы происходит после выделения необходимой области ЭКГ курсором мыши по алгоритму (рис 3.7):

1) Считывание выделенной области;

2) Удаление шумов с помощью фильтра Гаусса;

3) Перевод изображения в градацию серого;

4) Сегментация изображения.

Рисунок 3.7. Этапы обработки ЭКГ

Для работы программы необходимо выделить минимум 2 R-зубца. Если данное условие не было выполнено, появляется сообщение об ошибке (рис. 3.8).

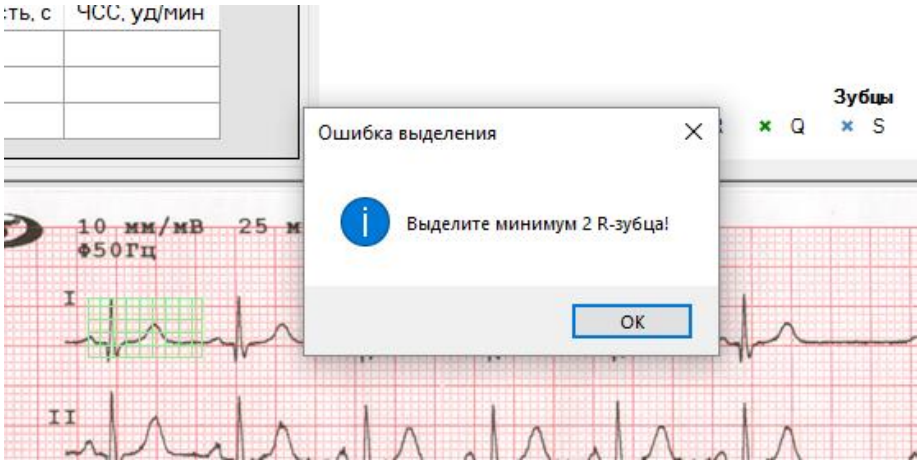

Рисунок 3.8. Сообщение ошибки выделения.

В результате корректной работы программы, после выделения как минимум двух R-зубцов на ЭКГ, в таблице в блоке ЧСС появятся значения, определенные программой (рис. 3.9-3.11).

Рисунок 3.9. Результаты работы программы.

Рисунок 3.10. Результаты работы программы.

Рисунок 3.11. Результаты работы программы.

3.2 Эксперименты и результаты

Общеустановленным методом оценивания качественности и верности алгоритма определения сигнала ЭКГ представляется проведение параллели между результатом его функционирования с аннотациями, исполненными квалифицированными кардиологами. Итоги в аннотациях считаются стопроцентно достоверными, и на основании их соизмерения с итогами

Чтобы оценить особенность характеристики работы алгоритма определения сигнала ЭКГ применяются два коэффициента: коэффициент чувствительности и коэффициент прогностичности положительного итога (или положительная прогностичность).

Чувствительность алгоритма Se – это способность алгоритма давать правильный результат. Определяется как доля истинно положительных случаев обнаружения точки среди всех фактически положительных случаев.

Положительная прогностичность P – это вероятность фактического наличия характерной точки при положительном результате ее обнаружения. Определяется как доля истинно положительных случаев среди всех положительных случаев обнаружения [5].

ЗАКЛЮЧЕНИЕ

Одной из наиболее важных задач кардиологии является проведение диагностики сердечно-сосудистой системы человека, так как именно заболеваниями в данной сфере обуславливаются основные причины смертности людей в трудоспособном возрасте. Объективное оценивание и прогнозирование состояния сердечно-сосудистой системы является одним из приоритетных направлений, связанных со здоровьем нации. На сегодняшний день к наиболее распространенным методам диагностики работы сердечно-сосудистой системы человека относится электрокардиограмма (ЭКГ). Однако, существующие средства мониторинга не дают однозначной интерпретации результатов ЭКГ при обращении любого пациента, что обусловливает необходимость разработки и совершенствования методов и алгоритмов оценивания изображений ЭКГ.

Использование информационных технологий при обработке ЭКГ используется при кардиологических исследованиях все чаще. Довольно часто применяемые технологии не могут обеспечить требуемой достоверности результатов диагностики, что обусловлено недостатками существующих алгоритмов распознавания информативных фрагментов ЭКГ.

Предложенные в магистерской диссертации процедура, метод и алгоритмы при их внедрении обеспечат повышение точности и надёжности формируемых диагностических заключений, что в дальнейшем позволит повысить эффективность диагностики и лечения патологий сердечно-сосудистой системы человека.

В процессе выполнения исследования был решен ряд задач.

1. Проведено аналитическое исследование предметной области, включая анализ современного состояния как методов, так и аппаратно-программного обеспечения обработки графиков ЭКС.

2. Проведен обзор современного состояния рынка программных продуктов.

3. Выявлены недостатки используемых в настоящее время в медицинских приборах методов и алгоритмов выявления возможных отклонений в работе сердечно-сосудистой системы на основе анализа данных, полученных ЭКГ.

4. Разработан алгоритмы, процедура и метод анализа изображений ЭКГ.

5. Спроектировано и программно-реализовано средство, обеспечивающее повышение точности и надёжности решения задач автоматического анализа изображений ЭКГ с целью получения значимой для диагностики информации о состоянии сердечно-сосудистой системы человека.

6. Проведено тестирование на реальных данных, по результатам которого можно сделать вывод, что программа имеет достаточную работоспособность и может использоваться по назначению.

Таким образом, все поставленные задачи решены и цель магистерской диссертации достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Харченко, В.В. Автоматизированная система анализа и интерпретации ЭКГ/ Харченко В.В., Дубровин В.И., Твердохлеб Ю.В. -Запорожье, Украина: Прогрессивные информационные технологии, 2014. – 156 с.
2. Калиниченко, А.Н. Компьютерные методы автоматического анализа ЭКГ в системах кардиологического наблюдения/ Санкт-Петербург, Приборы, системы и изделия медицинского назначения, 2008. – 205 с.
3. Аронсон, Ф. Наглядная кардиология [Текст] /The Cardiovascular System at a Glance / Ф. Аронсон, Дж. Вард, Г. Винер ; пер. с англ. ; под ред. С. Л. Дземешкевича. - Москва : ГЭОТАР-Медиа, 2011. - 120 с.
4. Арутюнов, Г. П. Терапия факторов риска сердечно-сосудистых заболеваний / Г. П. Арутюнов. - Москва : ГЭОТАР-Медиа, 2010. - 672 с.
5. Рудаков П.И, Сафонов В.И. Обработка сигналов и изображений Matlab 5.x. – Диалог-МИФИ. 2000. – 232 с.
6. Кляр Б. Цифровая связь. Теоретические основы и практическое применение: Пер. с англ. - М.: Издательский дом "Вильямс", 2003. – 135 с.
7. Воробьев, С.Н. Цифровая обработка сигналов: Учебник для студентов учреждений высшего профессионального образования / С.Н. Воробьев. - М.: ИЦ Академия, 2013. – 343 с.
8. . Давей, П. Наглядная ЭКГ / ECG at a Glance / П. Давей ; пер. с англ. ; под ред. М. В. Писарева. - Москва : ГЭОТАР-Медиа, 2011. - 168 с.
9. Юдин, Д.Б. Задачи и методы линейного программирования: Математические основы и практические задачи / Д.Б. Юдин, Е.Г. Гольштейн. - М.: КД Либроком, 2010. – 248 с.
10. Солонина, А.И. Цифровая обработка сигналов и MATLAB: Учебное пособие / А.И. Солонина, Д.М. Клионский, Т.В. Меркучева. - СПб.: БХВ-Петербург, 2013. – 168 с.

11. Зиборов, В. MS Visual C++ 2010 в среде .NET / В. Зиборов. - М.: Питер, 2012. – 263 с.
12. Понамарев, В. Программирование на C++/C# в Visual Studio .NET 2003 / В. Понамарев. - М.: БХВ-Петербург, 2015. – 302 с.
13. Титов К.В. Компьютерные технологии в вопросах изучения и решения задач интегральных преобразований и операционного исчисления: Учебное пособие по курсу "Спецглавы высшей математики". М.: Изд-во МГТУ им. Н.Э. Баумана. 2001г. – 352 с.
14. Гонсалес Р., Вудс Р. Цифровая обработка изображений / Пер. с англ. под ред. П.А. Чочиа – М.: ТЕХНОСФЕРА. 2005. – 174 с.
15. Игошин, В.И. Теория алгоритмов: Учебное пособие / В.И. Игошин. - М.: ИНФРА-М, 2013. – 256 с.
16. Тьюки Дж. Анализ результатов наблюдений. М. : Мир, 2008. – 175 с.
17. Шапкин, А.С. Математические методы и модели исследования операций: Учебник / А.С. Шапкин, В.А. Шапкин. - М.: Дашков и К, 2013. – 185 с.
18. Физика визуализации изображений в медицине: в 2–х томах. Т. 2: Пер. С англ. / Под ред. С. Уэбба. – М.: Мир, 2012. – 523 с.
19. Глотова, М. Ю. Математическая обработка информации. Учебник и практикум / М.Ю. Глотова, Е.А. Самохвалова. - М.: Юрайт, 2017. – 249 с.
20. Ярославский Л.П. Обработка изображений в медицинской интроскопии / Цифровая оптика в медицинской интроскопии. – М.: ИППИ РАН, 2015. – 225 с.
21. Нестерук В.Ф. Принцип дуальности при нелинейных безынерционных преобразованиях изображений // Труды ГОИ им. С.И. Вавилова. – 2003. – 308 с.
22. Агальцов, В.П. Математические методы в программировании: Учебник / В.П. Агальцов, И.В. Волдайская. - М.: ИД ФОРУМ, 2013. – 237 с.

23. Журавлев Ю.И., Калилов М.М., Гуляганов Ш.Е. Алгоритмы вычисления оценок и их применение. – Ташкент: Фан, 1974. – 251 с.
24. Латышенко, К.П. Автоматизация измерений, испытаний и контроля / К.П. Латышенко. - М.: МГУИЭ, 2006. – 187 с.
25. В.А. Скляров. Язык С++ и объектно-ориентированное программирование: Справочное издание. - Минск: Вышэйшая школа, 1997. – 335 с.
26. М. Эллис, Б. Строуструп. Справочное руководство по языку С++ с комментариями: Пер. с англ. - Москва: Мир, 1992. – 453 с.
27. Гупал, В.М. Математические методы анализа и распознавания аудиовизуальной информации: Монография / В.М. Гупал. - М.: ИЦ РИОР, НИЦ ИНФРА-М, 2012. – 241 с.
28. Курилова, А.В. Ввод и обработка цифровой информации. Практикум. Учебное пособие / А.В. Курилова. - М.: Академия (Academia), 2016. – 173 с.
29. Мячев, А. А. Мини- и микроэвм систем обработки информации / А.А. Мячев. - М.: Энергоатомиздат, 2017. – 205 с.
30. Денисенко, А.Н. Компьютерная обработка информации / А.Н. Денисенко. - М.: Медпрактика-М, 2017. – 144 с.
31. Дюк Вячеслав, Эмануэль Владимир Информационные технологии в медико-биологических исследованиях. СПб.:Питер, 2003. – 360 с.
32. Ахо А.В., Рави С, Ульман Дж.Д. Компиляторы: принципы, технологии и инструментарий — М.: Издательский дом "Вильямс", 2008 г. – 365 с.
33. Клюев, А.С. Автоматизация настройки систем управления / А.С. Клюев, В.Я. Ротач, В.Ф. Кузищин. - М.: Альянс, 2015. – 272 с.
34. Лбов Г.С. Анализ данных и знаний // Учебное пособие. Издательство НГТУ. Новосибирск 2001. – 331 с.
35. Лбов Г.С., Бериков В.Б. Устойчивость решающих функций в задачах распознавания образов и анализа разнотипной информации. - Новосибирск: Изд-во Ин-та математики, 2005. – 246 с.

36. Coulam C.M. Erickson J.J. and Gibbs S.J. Image and equipment considerations in conventional tomography // The Physical Basis of Medical Imaging ed. C.M. Coulam, J.J. Erickson and A.E. James (New York: Appleton – Century – Crofts). – 1981. – 412 с.
37. Мартынова А. С. Кардиология,основанная на доказательствах . Evidence – Based Cardiology. Second edition PDA. // Мартынова А.С., Корсакова Н.Г., Остроумова О.Т., РНКК. Москва 2005. – 277 с.
38. Кузнецов, С. И. Артериальная гипертония и артериальная гипотония: инновации комбинированной терапии: науч.-метод. пособие / С. И. Кузнецов, П. И. Романчук, Г. Г. Шишин ; Минздравсоцразвития СО, ГБОУ ВПО "СамГМУ", ГУЗ СО "Гериатрический науч.-практ. центр". - Самара : Волга-Бизнес, 2011. – 415 с.
39. Люсов, В. А. ЭКГ при инфаркте миокарда: практ. рук. / В. А. Люсов, Н. А. Волов, И. Г. Гордеев. - Москва : ГЭОТАР-Медиа, 2008. – 346 с.
40. Современные эхокардиографические подходы к оценке гипертрофии миокарда и структурного состояния левого желудочка у больных артериальной гипертонией: метод. пособие / М. А. Саидова [и др.] ; ФА по здравоохранению и соц. развитию РФ, ФГУ Рос. кардиологический науч.-производ. комплекс. - Москва : [б. и.], 2007. – 536 с.
41. М.В. Абрамов Аппроксимации экспонентами временного кардиологического ряда на основе ЭКГ / «Клиническая информатика и телемедицина», 2012. – 89 с.

ПРИЛОЖЕНИЯ

Приложение 1.

Листинг программы.

```
#pragma once
#include <opencv2\opencv.hpp>
#include <opencv2\highgui\highgui_c.h>
#include <opencv2\imgproc\imgproc.hpp>

namespace processingECG {

	using namespace cv;
	using namespace std;
	using namespace System;
	using namespace System::Linq;
	using namespace System::ComponentModel;
	using namespace System::Collections;
	using namespace System::Collections::Generic;
	using namespace System::Windows::Forms;
	using namespace System::Windows::Forms::DataVisualization::Charting;
	//using namespace System::Windows::Media;
	//using namespace System::IO;
	//using namespace System::Data;
	using namespace System::Drawing;
	using namespace System::Drawing::Drawing2D;
	using namespace System::Drawing::Imaging;
	using namespace System::Runtime::InteropServices;

	/// <summary>
	/// Summary for MyForm
	/// </summary>
	public ref class MainForm : public System::Windows::Forms::Form
	{
	public:
		MainForm(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~MainForm()
		{
			if (components)
			{
				delete components;
			}
		}
	private: System::Windows::Forms::PictureBox^  PictureECG;
	private: System::Windows::Forms::Panel^  PanelPuctureECG;
	private: System::Windows::Forms::DataVisualization::Charting::Chart^  ChartECG;
	private: System::Windows::Forms::MenuStrip^  MainMenu;
	private: System::Windows::Forms::ToolStripMenuItem^  файлToolStripMenuItem;
	private: System::Windows::Forms::ToolStripMenuItem^  открытьToolStripMenuItem;


	private: System::Windows::Forms::ToolStripSeparator^  toolStripSeparator2;
```

```
        private: System::Windows::Forms::ToolStripMenuItem^  выходToolStripMenuItem;
        private: System::Windows::Forms::OpenFileDialog^  OpenFileDialog;
        private: System::Windows::Forms::TabPage^  tabPage2;
        private: System::Windows::Forms::GroupBox^  groupBox1;
        private: System::Windows::Forms::DataGridView^  DataCSS;
        private: System::Windows::Forms::TabPage^  PageECG3;
        private: System::Windows::Forms::GroupBox^  groupBox3;
        private: System::Windows::Forms::Label^  label3;
        private: System::Windows::Forms::NumericUpDown^  SpeedECG;
        private: System::Windows::Forms::TabPage^  PageECG1;
        private: System::Windows::Forms::GroupBox^  GroupBoxStep1;
        private: System::Windows::Forms::Label^  label2;
        private: System::Windows::Forms::NumericUpDown^  LenMillimeter;
        private: System::Windows::Forms::TabControl^  PagesECG;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  ColumnTypeCSS;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  ColumnTimeCSS;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  ColumnCSS;
        private:
            /// <summary>
            /// Required designer variable.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                System::Windows::Forms::DataVisualization::Charting::ChartArea^  chartArea1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::ChartArea());
                System::Windows::Forms::DataVisualization::Charting::Legend^  legend1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Legend());
                System::Windows::Forms::DataVisualization::Charting::Series^  series1 = (gcnew
System::Windows::Forms::DataVisualization::Charting::Series());
                System::ComponentModel::ComponentResourceManager^  resources = (gcnew
System::ComponentModel::ComponentResourceManager(MainForm::typeid));
                this->PictureECG = (gcnew System::Windows::Forms::PictureBox());
                this->PanelPuctureECG = (gcnew System::Windows::Forms::Panel());
                this->ChartECG = (gcnew System::Windows::Forms::DataVisualization::Charting::Chart());
                this->MainMenu = (gcnew System::Windows::Forms::MenuStrip());
                this->файлToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
                this->открытьToolStripMenuItem = (gcnew
System::Windows::Forms::ToolStripMenuItem());
                this->toolStripSeparator2 = (gcnew System::Windows::Forms::ToolStripSeparator());
                this->выходToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
                this->OpenFileDialog = (gcnew System::Windows::Forms::OpenFileDialog());
                this->tabPage2 = (gcnew System::Windows::Forms::TabPage());
                this->groupBox1 = (gcnew System::Windows::Forms::GroupBox());
                this->DataCSS = (gcnew System::Windows::Forms::DataGridView());
                this->ColumnTypeCSS = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
                this->ColumnTimeCSS = (gcnew
System::Windows::Forms::DataGridViewTextBoxColumn());
                this->ColumnCSS = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
                this->PageECG3 = (gcnew System::Windows::Forms::TabPage());
                this->groupBox3 = (gcnew System::Windows::Forms::GroupBox());
                this->label3 = (gcnew System::Windows::Forms::Label());
                this->SpeedECG = (gcnew System::Windows::Forms::NumericUpDown());
                this->PageECG1 = (gcnew System::Windows::Forms::TabPage());
                this->GroupBoxStep1 = (gcnew System::Windows::Forms::GroupBox());
                this->label2 = (gcnew System::Windows::Forms::Label());
                this->LenMillimeter = (gcnew System::Windows::Forms::NumericUpDown());
                this->PagesECG = (gcnew System::Windows::Forms::TabControl());
```

```
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->PictureECG))-
>BeginInit();
                this->PanelPuctureECG->SuspendLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->ChartECG))-
>BeginInit();
                this->MainMenu->SuspendLayout();
                this->groupBox1->SuspendLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->DataCSS))-
>BeginInit();
                this->PageECG3->SuspendLayout();
                this->groupBox3->SuspendLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->SpeedECG))-
>BeginInit();
                this->PageECG1->SuspendLayout();
                this->GroupBoxStep1->SuspendLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->LenMillimeter))-
>BeginInit();
                this->PagesECG->SuspendLayout();
                this->SuspendLayout();
                //
                // PictureECG
                //
                this->PictureECG->BackColor = System::Drawing::SystemColors::ControlLight;
                this->PictureECG->Cursor = System::Windows::Forms::Cursors::Cross;
                this->PictureECG->Location = System::Drawing::Point(3, 3);
                this->PictureECG->Name = L"PictureECG";
                this->PictureECG->Size = System::Drawing::Size(646, 233);
                this->PictureECG->SizeMode = System::Windows::Forms::PictureBoxSizeMode::AutoSize;
                this->PictureECG->TabIndex = 0;
                this->PictureECG->TabStop = false;
                this->PictureECG->MouseDown += gcnew
System::Windows::Forms::MouseEventHandler(this, &MainForm::PictureECG_MouseDown);
                this->PictureECG->MouseMove += gcnew
System::Windows::Forms::MouseEventHandler(this, &MainForm::PictureECG_MouseMove);
                this->PictureECG->MouseUp += gcnew System::Windows::Forms::MouseEventHandler(this,
&MainForm::PictureECG_MouseUp);
                //
                // PanelPuctureECG
                //
                this->PanelPuctureECG->AutoScroll = true;
                this->PanelPuctureECG->BackColor = System::Drawing::SystemColors::ControlLight;
                this->PanelPuctureECG->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;
                this->PanelPuctureECG->Controls->Add(this->PictureECG);
                this->PanelPuctureECG->Location = System::Drawing::Point(16, 284);
                this->PanelPuctureECG->Name = L"PanelPuctureECG";
                this->PanelPuctureECG->Size = System::Drawing::Size(1006, 258);
                this->PanelPuctureECG->TabIndex = 1;
                //
                // ChartECG
                //
                chartArea1->AxisX->IsMarginVisible = false;
                chartArea1->AxisX->LabelStyle->Enabled = false;
                chartArea1->AxisX->LineWidth = 0;
                chartArea1->AxisX->MajorGrid->LineColor = System::Drawing::Color::Gainsboro;
                chartArea1->AxisX->MajorTickMark->Enabled = false;
                chartArea1->AxisY->IsMarginVisible = false;
                chartArea1->AxisY->LabelStyle->Enabled = false;
                chartArea1->AxisY->LineWidth = 0;
                chartArea1->AxisY->MajorGrid->LineColor = System::Drawing::Color::Gainsboro;
                chartArea1->AxisY->MajorTickMark->Enabled = false;
                chartArea1->Name = L"ChartAreaECG";
                this->ChartECG->ChartAreas->Add(chartArea1);
                legend1->Alignment = System::Drawing::StringAlignment::Center;
                legend1->Docking =
System::Windows::Forms::DataVisualization::Charting::Docking::Bottom;
```

```
                legend1->Name = L"LegendBattlement";
                legend1->Title = L"Зубцы";
                this->ChartECG->Legends->Add(legend1);
                this->ChartECG->Location = System::Drawing::Point(363, 41);
                this->ChartECG->Name = L"ChartECG";
                series1->ChartArea = L"ChartAreaECG";
                series1->ChartType =
System::Windows::Forms::DataVisualization::Charting::SeriesChartType::Spline;
                series1->Color = System::Drawing::Color::Black;
                series1->IsVisibleInLegend = false;
                series1->Legend = L"LegendBattlement";
                series1->Name = L"SeriesECG";
                this->ChartECG->Series->Add(series1);
                this->ChartECG->Size = System::Drawing::Size(659, 231);
                this->ChartECG->TabIndex = 0;
                this->ChartECG->Text = L"chart1";
                //
                // MainMenu
                //
                this->MainMenu->Items->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(1) { this->файлToolStripMenuItem });
                this->MainMenu->Location = System::Drawing::Point(0, 0);
                this->MainMenu->Name = L"MainMenu";
                this->MainMenu->Size = System::Drawing::Size(1034, 24);
                this->MainMenu->TabIndex = 3;
                this->MainMenu->Text = L"menuStrip1";
                //
                // файлToolStripMenuItem
                //
                this->файлToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array<
System::Windows::Forms::ToolStripItem^  >(3) {
                        this->открытьToolStripMenuItem,
                                this->toolStripSeparator2, this->выходToolStripMenuItem
                });
                this->файлToolStripMenuItem->Name = L"файлToolStripMenuItem";
                this->файлToolStripMenuItem->Size = System::Drawing::Size(48, 20);
                this->файлToolStripMenuItem->Text = L"&Файл";
                //
                // открытьToolStripMenuItem
                //
                this->открытьToolStripMenuItem->Image =
(cli::safe_cast<System::Drawing::Image^>(resources->GetObject(L"открытьToolStripMenuItem.Image")));
                this->открытьToolStripMenuItem->ImageTransparentColor =
System::Drawing::Color::Magenta;
                this->открытьToolStripMenuItem->Name = L"открытьToolStripMenuItem";
                this->открытьToolStripMenuItem->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control |
System::Windows::Forms::Keys::O));
                this->открытьToolStripMenuItem->Size = System::Drawing::Size(164, 22);
                this->открытьToolStripMenuItem->Text = L"&Открыть";
                this->открытьToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MainForm::открытьToolStripMenuItem_Click);
                //
                // toolStripSeparator2
                //
                this->toolStripSeparator2->Name = L"toolStripSeparator2";
                this->toolStripSeparator2->Size = System::Drawing::Size(161, 6);
                //
                // выходToolStripMenuItem
                //
                this->выходToolStripMenuItem->Name = L"выходToolStripMenuItem";
                this->выходToolStripMenuItem->Size = System::Drawing::Size(164, 22);
                this->выходToolStripMenuItem->Text = L"Вы&ход";
                this->выходToolStripMenuItem->Click += gcnew System::EventHandler(this,
&MainForm::выходToolStripMenuItem_Click);
```

```
                //
                // OpenFileDialog
                //
                this->OpenFileDialog->FileName = L"Выберите изображение ЭКГ";
                this->OpenFileDialog->Filter = L"JPG|*.jpg;*.jpeg";
                //
                // tabPage2
                //
                this->tabPage2->Location = System::Drawing::Point(0, 0);
                this->tabPage2->Name = L"tabPage2";
                this->tabPage2->Padding = System::Windows::Forms::Padding(3);
                this->tabPage2->Size = System::Drawing::Size(200, 100);
                this->tabPage2->TabIndex = 0;
                this->tabPage2->Text = L"tabPage2";
                //
                // groupBox1
                //
                this->groupBox1->Controls->Add(this->DataCSS);
                this->groupBox1->Font = (gcnew System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
System::Drawing::FontStyle::Regular, System::Drawing::GraphicsUnit::Point,
                        static_cast<System::Byte>(204)));
                this->groupBox1->Location = System::Drawing::Point(16, 144);
                this->groupBox1->Name = L"groupBox1";
                this->groupBox1->Size = System::Drawing::Size(341, 134);
                this->groupBox1->TabIndex = 7;
                this->groupBox1->TabStop = false;
                this->groupBox1->Text = L"ЧСС ";
                //
                // DataCSS
                //
                this->DataCSS->AllowUserToAddRows = false;
                this->DataCSS->BackgroundColor = System::Drawing::SystemColors::ControlLight;
                this->DataCSS->ColumnHeadersHeightSizeMode =
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
                this->DataCSS->Columns->AddRange(gcnew cli::array<
System::Windows::Forms::DataGridViewColumn^ >(3) {
                        this->ColumnTypeCSS,
                                this->ColumnTimeCSS, this->ColumnCSS
                });
                this->DataCSS->Location = System::Drawing::Point(10, 25);
                this->DataCSS->Name = L"DataCSS";
                this->DataCSS->ReadOnly = true;
                this->DataCSS->RowHeadersVisible = false;
                this->DataCSS->RowTemplate->Resizable =
System::Windows::Forms::DataGridViewTriState::False;
                this->DataCSS->Size = System::Drawing::Size(325, 103);
                this->DataCSS->TabIndex = 0;
                //
                // ColumnTypeCSS
                //
                this->ColumnTypeCSS->HeaderText = L"";
                this->ColumnTypeCSS->Name = L"ColumnTypeCSS";
                this->ColumnTypeCSS->ReadOnly = true;
                this->ColumnTypeCSS->Resizable =
System::Windows::Forms::DataGridViewTriState::False;
                this->ColumnTypeCSS->SortMode =
System::Windows::Forms::DataGridViewColumnSortMode::NotSortable;
                this->ColumnTypeCSS->Width = 60;
                //
                // ColumnTimeCSS
                //
                this->ColumnTimeCSS->HeaderText = L"Длительность, с";
                this->ColumnTimeCSS->Name = L"ColumnTimeCSS";
                this->ColumnTimeCSS->ReadOnly = true;
```

```
                this->ColumnTimeCSS->Resizable =
System::Windows::Forms::DataGridViewTriState::False;
                this->ColumnTimeCSS->SortMode =
System::Windows::Forms::DataGridViewColumnSortMode::NotSortable;
                this->ColumnTimeCSS->Width = 120;
                //
                // ColumnCSS
                //
                this->ColumnCSS->HeaderText = L"ЧСС, уд/мин";
                this->ColumnCSS->Name = L"ColumnCSS";
                this->ColumnCSS->ReadOnly = true;
                this->ColumnCSS->Resizable = System::Windows::Forms::DataGridViewTriState::False;
                this->ColumnCSS->SortMode =
System::Windows::Forms::DataGridViewColumnSortMode::NotSortable;
                this->ColumnCSS->Width = 95;
                //
                // PageECG3
                //
                this->PageECG3->Controls->Add(this->groupBox3);
                this->PageECG3->Location = System::Drawing::Point(4, 22);
                this->PageECG3->Name = L"PageECG3";
                this->PageECG3->Padding = System::Windows::Forms::Padding(3);
                this->PageECG3->Size = System::Drawing::Size(333, 63);
                this->PageECG3->TabIndex = 2;
                this->PageECG3->Text = L"2. Скорость записи ЭКГ";
                this->PageECG3->UseVisualStyleBackColor = true;
                //
                // groupBox3
                //
                this->groupBox3->Controls->Add(this->label3);
                this->groupBox3->Controls->Add(this->SpeedECG);
                this->groupBox3->Location = System::Drawing::Point(6, 6);
                this->groupBox3->Name = L"groupBox3";
                this->groupBox3->RightToLeft = System::Windows::Forms::RightToLeft::No;
                this->groupBox3->Size = System::Drawing::Size(321, 50);
                this->groupBox3->TabIndex = 5;
                this->groupBox3->TabStop = false;
                this->groupBox3->Text = L"Введите скорость записи ЭКГ";
                //
                // label3
                //
                this->label3->AutoSize = true;
                this->label3->Location = System::Drawing::Point(6, 21);
                this->label3->Name = L"label3";
                this->label3->RightToLeft = System::Windows::Forms::RightToLeft::No;
                this->label3->Size = System::Drawing::Size(88, 13);
                this->label3->TabIndex = 3;
                this->label3->Text = L"Скорость, мм/с";
                //
                // SpeedECG
                //
                this->SpeedECG->Location = System::Drawing::Point(263, 19);
                this->SpeedECG->Minimum = System::Decimal(gcnew cli::array< System::Int32 >(4) { 20,
0, 0, 0 });
                this->SpeedECG->Name = L"SpeedECG";
                this->SpeedECG->Size = System::Drawing::Size(52, 20);
                this->SpeedECG->TabIndex = 0;
                this->SpeedECG->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 50, 0, 0,
0 });
                //
                // PageECG1
                //
                this->PageECG1->Controls->Add(this->GroupBoxStep1);
                this->PageECG1->Location = System::Drawing::Point(4, 22);
                this->PageECG1->Name = L"PageECG1";
```

```
this->PageECG1->Padding = System::Windows::Forms::Padding(3);
this->PageECG1->Size = System::Drawing::Size(333, 63);
this->PageECG1->TabIndex = 0;
this->PageECG1->Text = L"1. Размеры миллиметровой клеточки";
this->PageECG1->UseVisualStyleBackColor = true;
//
// GroupBoxStep1
//
this->GroupBoxStep1->Controls->Add(this->label2);
this->GroupBoxStep1->Controls->Add(this->LenMillimeter);
this->GroupBoxStep1->Location = System::Drawing::Point(6, 6);
this->GroupBoxStep1->Name = L"GroupBoxStep1";
this->GroupBoxStep1->RightToLeft = System::Windows::Forms::RightToLeft::No;
this->GroupBoxStep1->Size = System::Drawing::Size(321, 50);
this->GroupBoxStep1->TabIndex = 1;
this->GroupBoxStep1->TabStop = false;
this->GroupBoxStep1->Text = L"Выделите полный квадрат 5x5 клеточек";
//
// label2
//
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(6, 21);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(145, 13);
this->label2->TabIndex = 3;
this->label2->Text = L"Длина и ширина клетки, px";
//
// LenMillimeter
//
this->LenMillimeter->DecimalPlaces = 2;
this->LenMillimeter->Location = System::Drawing::Point(263, 19);
this->LenMillimeter->Maximum = System::Decimal(gcnew cli::array< System::Int32 >(4) {
10000, 0, 0, 0 });
this->LenMillimeter->Name = L"LenMillimeter";
this->LenMillimeter->Size = System::Drawing::Size(52, 20);
this->LenMillimeter->TabIndex = 0;
this->LenMillimeter->Value = System::Decimal(gcnew cli::array< System::Int32 >(4) { 1, 0,
0, 0 });
//
// PagesECG
//
this->PagesECG->Controls->Add(this->PageECG1);
this->PagesECG->Controls->Add(this->PageECG3);
this->PagesECG->Location = System::Drawing::Point(16, 49);
this->PagesECG->Name = L"PagesECG";
this->PagesECG->SelectedIndex = 0;
this->PagesECG->Size = System::Drawing::Size(341, 89);
this->PagesECG->TabIndex = 6;
//
// MainForm
//
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->AutoValidate = System::Windows::Forms::AutoValidate::EnablePreventFocusChange;
this->ClientSize = System::Drawing::Size(1034, 554);
this->Controls->Add(this->groupBox1);
this->Controls->Add(this->PagesECG);
this->Controls->Add(this->PanelPuctureECG);
this->Controls->Add(this->ChartECG);
this->Controls->Add(this->MainMenu);
this->MainMenuStrip = this->MainMenu;
this->Name = L"MainForm";
this->Text = L"Компьютерный анализ ЧСС";
this->Load += gcnew System::EventHandler(this, &MainForm::MainForm_Load);
this->Resize += gcnew System::EventHandler(this, &MainForm::MainForm_Resize);
```

```
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->PictureECG))-
>EndInit();
                this->PanelPuctureECG->ResumeLayout(false);
                this->PanelPuctureECG->PerformLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->ChartECG))->EndInit();
                this->MainMenu->ResumeLayout(false);
                this->MainMenu->PerformLayout();
                this->groupBox1->ResumeLayout(false);
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->DataCSS))->EndInit();
                this->PageECG3->ResumeLayout(false);
                this->groupBox3->ResumeLayout(false);
                this->groupBox3->PerformLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->SpeedECG))-
>EndInit();
                this->PageECG1->ResumeLayout(false);
                this->GroupBoxStep1->ResumeLayout(false);
                this->GroupBoxStep1->PerformLayout();
                (cli::safe_cast<System::ComponentModel::ISupportInitialize^>(this->LenMillimeter))-
>EndInit();
                this->PagesECG->ResumeLayout(false);
                this->ResumeLayout(false);
                this->PerformLayout();

        }
#pragma endregion
    Rectangle region;
    private: Mat ConvertDrawImageToCVMat(Image ^image, Rectangle area) {
        Bitmap ^bitmap = safe_cast<Bitmap^>(image);
        int w = area.Width;
        int h = area.Height;
        BitmapData^ bitmapData = bitmap->LockBits(area, ImageLockMode::ReadWrite, bitmap-
>PixelFormat);
        Mat thisimage;// (cv::Size(image->Width, image->Height), CV_8UC3, bmpdata->Scan0.ToPointer(),
Mat::AUTO_STEP);
        try
        {
            IntPtr ptr = bitmapData->Scan0; // Получить адрес первой строки.
            // Объявляем массив для хранения байтов растрового изображения.
            // Этот код специфичен для растрового изображения с 24 бит на пиксель.
            int bytes = Math::Abs(bitmapData->Stride) * h;
            cli::array<Byte>^rgbValues = gcnew cli::array<Byte>(bytes);
            // Скопируйте значения RGB в массив.
            Marshal::Copy(ptr, rgbValues, 0, bytes);
            return Mat(h, w, CV_8UC3, (void *)ptr, Math::Abs(bitmapData->Stride));
        }
        finally { bitmap->UnlockBits(bitmapData); }

    }

    private: Void MainForm_Load(System::Object^  sender, System::EventArgs^  e) {
        //PictureECG->BackgroundImage = Image::FromFile("img/d62e2efdf13f.jpg");
        DataCSS->Rows->Clear();
        DataCSS->Rows->Add("Макс.");
        DataCSS->Rows->Add("Средн.");
        DataCSS->Rows->Add("Мин.");

        for (int i = 0; i < 5; i++) {
            System::String ^name = "Series";
            System::String ^legendText = "";
            Color color;
            switch (i)
            {
            case 0: { legendText += "R"; name += legendText; color = Color::Red; break; }
            case 1: { legendText += "Q"; name += legendText; color = Color::Green; break; }
            case 2: { legendText += "S"; name += legendText; color = Color::SteelBlue; break; }
```

```
                        case 3: { legendText += "P"; name += legendText; color = Color::Orange; break; }
                        default: { legendText += "T"; name += legendText; color = Color::Blue; }
                        }
                        Series ^series = gcnew Series(name);
                        series->Color = color;
                        series->ChartType = SeriesChartType::Point;
                        series->Legend = "LegendBattlement";
                        series->LegendText = legendText;
                        series->MarkerSize = 10;
                        //series->MarkerBorderWidth = 1;
                        //series->MarkerBorderColor = Color::Black;
                        series->MarkerStyle = MarkerStyle::Cross;
                        ChartECG->Series->Add(series);
                }


        }

        private: Void ProcessingImage(Mat src) {
                cvDestroyAllWindows();
                //namedWindow("src", WINDOW_AUTOSIZE); imshow("src", src);
                if (src.empty())
                        return;
                Mat srcFilter;
                Mat srcGray;
                Mat srcSgm;

                cv::Size coreFilter = cv::Size(1, 1);
                switch (PagesECG->SelectedIndex)
                {
                case 0: { coreFilter = cv::Size(1, 1); break; }
                case 1: { coreFilter = cv::Size(5, 5); break; }
                default:;
                }

                //bilateralFilter(src, srcFilter, 2, 3.0, 3.0); // (двустороннее) сглаживание
                GaussianBlur(src, srcFilter, coreFilter, 0, 0); // фильтрация по Гауссу
                //medianBlur(src, srcFilter, 1); // медианная фильтрация
                //namedWindow("srcFilter", WINDOW_AUTOSIZE); imshow("srcFilter", srcFilter);
                cvtColor(srcFilter, srcGray, CV_BGR2GRAY);
                //namedWindow("srcGray", WINDOW_AUTOSIZE); imshow("srcGray", srcGray);

                // считывание точек из cv::Mat
                List<int> ^data = gcnew List<int>();
                List<System::Drawing::Point> ^dataXY = gcnew List<System::Drawing::Point>();

                int minY = 0;
                // подбор параметров сегментации так, чтобы можно было однозначно выделить ЭКГ
                for (double thresh = 100.0; thresh < 256.0 && minY == 0; thresh++) {
                        dataXY->Clear();
                        threshold(srcGray, srcSgm, thresh, 255.0, CV_THRESH_BINARY);
                        //namedWindow("imageSegmentation", WINDOW_AUTOSIZE);
imshow("imageSegmentation", srcSgm);
                        for (int x = 0, pY = 0; x < srcSgm.cols; x++, pY = 0) {
                                for (int y = 0; y < srcSgm.rows && pY == 0; y++) {
                                        uchar *ptr = (uchar*)(srcSgm.data + y * srcSgm.step);
                                        if (ptr[x] == 0) pY = srcSgm.rows - y;
                                }
                                dataXY->Add(System::Drawing::Point(x, pY));
                        }

                        // удаление лишних точек ЭКГ диаграммы
                        for (int i = 0, x = 0, count = 1; i < dataXY->Count - 1; i++) {
                                if (dataXY[i].Y == dataXY[i + 1].Y) {
                                        x += dataXY[i + 1].X;
```

```
                                        dataXY->RemoveAt(i + 1);
                                        i--;
                                        count++;
                                }
                                else {
                                        x += dataXY[i].X;
                                        dataXY[i] = System::Drawing::Point(x / count, dataXY[i].Y);
                                        x = 0;
                                        count = 1;
                                }
                        }
                        data->Clear();
                        for (int i = 0; i < dataXY->Count; i++)
                                data->Add(dataXY[i].Y);
                        minY = Enumerable::Min(data);

                        if (thresh == 255.0 && data->Count == 0) {
                                MessageBox::Show("Выделите минимум 2 R-зубца!", "Ошибка выделения",
MessageBoxButtons::OK, MessageBoxIcon::Information);
                                return;
                        }
                }

                /*
                Mat imgSegResize;
                resize(srcSgm, imgSegResize, cv::Size(srcSgm.cols * 4, srcSgm.rows * 4), 0.0, 0.0, INTER_CUBIC);
                namedWindow("imgSegResize", WINDOW_AUTOSIZE); imshow("imgSegResize", imgSegResize);
                imgSegResize.release();
                */

                switch (PagesECG->SelectedIndex)
                {
                case 0: { // определение длины миллиметровой клеточки
                        int posLineUp = 0;
                        int posLineDown = 0;

                        for (double thresh = 50.0; thresh < 256.0 && posLineUp == 0; thresh += 0.5) {
                                threshold(srcGray, srcSgm, thresh, 255.0, CV_THRESH_BINARY);
                                for (int y = 0; y < srcSgm.rows / 2; y++) {
                                        int x = 0;
                                        uchar *ptr = (uchar*)(srcSgm.data + y * srcSgm.step);
                                        for (x = 0; x < srcSgm.cols; x++)
                                                if (ptr[x] != 0) break;
                                        if (x == srcSgm.cols) {       posLineUp = y; break; }
                                }
                        }

                        for (double thresh = 50.0; thresh < 256.0 && posLineDown == 0; thresh += 0.5) {
                                threshold(srcGray, srcSgm, thresh, 255.0, CV_THRESH_BINARY);
                                for (int y = srcSgm.rows - 1; y >= srcSgm.rows / 2; y--) {
                                        int x = 0;
                                        uchar *ptr = (uchar*)(srcSgm.data + y * srcSgm.step);
                                        for (x = 0; x < srcSgm.cols; x++)
                                                if (ptr[x] != 0) break;
                                        if (x == srcSgm.cols) { posLineDown = y; break; }
                                }
                        }

                        if (posLineUp != 0 && posLineDown != 0) {
                                LenMillimeter->Value = Convert::ToDecimal(double(posLineDown - posLineUp) /
5.0);
                                ChartECG->ChartAreas["ChartAreaECG"]->AxisX->MajorGrid->Interval =
Convert::ToDouble(LenMillimeter->Value);
                                ChartECG->ChartAreas["ChartAreaECG"]->AxisY->MajorGrid->Interval =
Convert::ToDouble(LenMillimeter->Value);
```

```
                                        //namedWindow("srcSgm", WINDOW_AUTOSIZE); imshow("srcSgm", srcSgm);
                                }
                                else MessageBox::Show("Выделите квадрат 5x5 клеточек!", "Ошибка выделения",
MessageBoxButtons::OK, MessageBoxIcon::Information);
                                break;
                        }
                        case 1: { // поиск зубцов и расчет ЧСС

                                for (int i = 0, minY = Enumerable::Min(data); i < dataXY->Count; i++)
                                        dataXY[i] = System::Drawing::Point(dataXY[i].X, dataXY[i].Y - minY);

                                data->Clear();
                                for (int i = 0; i < dataXY->Count; i++)
                                        data->Add(dataXY[i].Y);
                                int max = Enumerable::Max(data);

                                // поиск зубцов R
                                List<System::Drawing::Point> ^R = gcnew List<System::Drawing::Point>();
                                int area = 5; // окресность точки локального максимума
                                List<List<System::Drawing::Point>^> ^temp = gcnew
List<List<System::Drawing::Point>^>();
                                for (int k = 1; k < max / 2; k++) {
                                        for (int i = area; i < dataXY->Count - area; i++) {
                                                if (max - dataXY[i].Y <= k) {
                                                        bool locMax = true;
                                                        for (int j = -area; j <= area && locMax; j++)
                                                                locMax = (dataXY[i + j].Y <= dataXY[i].Y) ? true :
false;
                                                        if (locMax)
                                                                R->Add(System::Drawing::Point(dataXY[i]));
                                                }
                                        }
                                        temp->Add(gcnew List<System::Drawing::Point>(R));
                                        R->Clear();
                                }

                                R = gcnew List<System::Drawing::Point>(temp[1]);
                                for (int i = 1, maxNextAmount = 1, maxLastAmount = 0; i < temp->Count; i++) {
                                        if (temp[i - 1]->Count == temp[i]->Count) maxNextAmount++;
                                        if (temp[i - 1]->Count != temp[i]->Count || i == temp->Count - 1) {
                                                if (maxNextAmount >= maxLastAmount) {
                                                        maxLastAmount = maxNextAmount;
                                                        R = gcnew List<System::Drawing::Point>(temp[i - 1]);
                                                }
                                                maxNextAmount = 1;
                                        }
                                }

                                if (R->Count == 1)
                                        MessageBox::Show("Выделите минимум 2 R-зубца!", "Ошибка выделения",
MessageBoxButtons::OK, MessageBoxIcon::Information);

                                // поиск зубцов Q
                                List<System::Drawing::Point> ^Q = gcnew List<System::Drawing::Point>();
                                for (int i = 0, j = 0; i < R->Count; i++, j = 0) {
                                        for (j = dataXY->IndexOf(R[i]); j >= 1 && dataXY[j - 1].Y < dataXY[j].Y; j--);
                                        Q->Add(dataXY[j]);
                                }

                                // поиск зубцов S
                                List<System::Drawing::Point> ^S = gcnew List<System::Drawing::Point>();
                                for (int i = 0, j = 0; i < R->Count; i++, j = 0) {
                                        for (j = dataXY->IndexOf(R[i]); j < dataXY->Count - 1 && dataXY[j].Y > dataXY[j
+ 1].Y; j++);
                                        S->Add(dataXY[j]);
```

```
                }

                // поиск зубцов P и T
                List<System::Drawing::Point> ^P = gcnew List<System::Drawing::Point>();
                List<System::Drawing::Point> ^T = gcnew List<System::Drawing::Point>();
                for (int i = 0; i < Q->Count + 1; i++) {
                        int start, count;
                        if (i == 0) { start = 0; count = dataXY->IndexOf(Q[i]) - start;      }
                        else if (i == Q->Count) { start = dataXY->IndexOf(S[i - 1]); count = data->Count -
start; }
                        else { start = dataXY->IndexOf(S[i - 1]); count = dataXY->IndexOf(Q[i]) - start; }

                        List<System::Drawing::Point> ^rangeXY = gcnew
List<System::Drawing::Point>(dataXY->GetRange(start, count));
                        List<System::Drawing::Point> ^locMaxPoint = gcnew
List<System::Drawing::Point>();
                        for (int j = area; j < rangeXY->Count - area; j++) {
                                bool locMax = true;
                                for (int k = -area; k <= area && locMax; k++)
                                        locMax = (rangeXY[j + k].Y <= rangeXY[j].Y) ? true : false;
                                if (locMax)
                                        locMaxPoint->Add(System::Drawing::Point(rangeXY[j]));
                        }
                        System::Drawing::Point battlementP;
                        System::Drawing::Point battlementT;
                        if (i < Q->Count) {
                                for (int j = 0, max = INT_MIN; j < locMaxPoint->Count; j++)
                                        if (max < locMaxPoint[j].Y && (max = locMaxPoint[j].Y))
                                                battlementP = locMaxPoint[j];

                                locMaxPoint->Remove(battlementP);

                        }
                        if (i > 0) {
                                for (int j = 0, max = INT_MIN; j < locMaxPoint->Count; j++)
                                        if (max < locMaxPoint[j].Y && (max = locMaxPoint[j].Y))
                                                battlementT = locMaxPoint[j];
                        }
                        if (0 < i && i < Q->Count && battlementT.X > battlementP.X) {
                                System::Drawing::Point point = battlementP;
                                battlementP = battlementT;
                                battlementT = point;
                        }
                        if (i < Q->Count) P->Add(battlementP);
                        if (i > 0) T->Add(battlementT);
                }

                int middle = 0;
                int min = INT_MAX;
                max = INT_MIN;
                for (int i = 1, diff = 0; i < R->Count; i++) {
                        diff = R[i].X - R[i - 1].X;
                        max = (max < diff) ? diff : max;
                        min = (min >= diff) ? diff : min;
                        middle += diff;
                }
                DataCSS->Rows->Clear();
                for (int i = 0; i < ChartECG->Series->Count; i++) ChartECG->Series[i]->Points->Clear();
                ChartECG->Legends["LegendBattlement"]->Enabled = false;
                if (R->Count - 1 != 0) {
                        ChartECG->Legends["LegendBattlement"]->Enabled = true;

                        for (int i = 0; i < dataXY->Count; i++)
                                ChartECG->Series["SeriesECG"]->Points->AddXY(dataXY[i].X,
dataXY[i].Y);
```

```
                                        for (int i = 0; i < P->Count; i++)
                                                ChartECG->Series["SeriesP"]->Points->AddXY(P[i].X, P[i].Y);
                                        for (int i = 0; i < Q->Count; i++)
                                                ChartECG->Series["SeriesQ"]->Points->AddXY(Q[i].X, Q[i].Y);
                                        for (int i = 0; i < R->Count; i++)
                                                ChartECG->Series["SeriesR"]->Points->AddXY(R[i].X, R[i].Y);

                                        for (int i = 0; i < S->Count; i++)
                                                ChartECG->Series["SeriesS"]->Points->AddXY(S[i].X, S[i].Y);
                                        for (int i = 0; i < T->Count; i++)
                                                ChartECG->Series["SeriesT"]->Points->AddXY(T[i].X, T[i].Y);

                                        middle /= R->Count - 1;
                                        double time = (max / Convert::ToDouble(LenMillimeter->Value)) /
Convert::ToDouble(SpeedECG->Value);
                                        DataCSS->Rows->Add("Макс.", Math::Round(time, 2), int(60.0 / time));
                                        time = (middle / Convert::ToDouble(LenMillimeter->Value)) /
Convert::ToDouble(SpeedECG->Value);
                                        DataCSS->Rows->Add("Средн.", Math::Round(time, 2), int(60.0 / time));
                                        time = (min / Convert::ToDouble(LenMillimeter->Value)) /
Convert::ToDouble(SpeedECG->Value);
                                        DataCSS->Rows->Add("Мин.", Math::Round(time, 2), int(60.0 / time));
                                }
                                break;
                        }
                        default:;
                        }

                        /*
                        Bitmap ^bitmap = gcnew Bitmap(imgSgm.cols, imgSgm.rows, 4 * imgSgm.rows,
System::Drawing::Imaging::PixelFormat::Format24bppRgb, IntPtr(imgSgm.data));

                        Bitmap ^bmp = gcnew Bitmap(bitmap);
                        MemoryStream ^ms = gcnew MemoryStream();
                        bmp->Save(ms, ImageFormat::Bmp);
                        System::Windows::Media::Imaging::BitmapImage ^img = gcnew
System::Windows::Media::Imaging::BitmapImage();
                        img->BeginInit();
                        ms->Seek(0, SeekOrigin::Begin);
                        img->StreamSource = ms;
                        img->EndInit();

                        ImageSource ^sc = (ImageSource^)img;
                        */

                        /*
                        System::Drawing::Imaging::PixelFormat
pixelFormat(System::Drawing::Imaging::PixelFormat::Format24bppRgb);
                        Bitmap ^bmpImg = gcnew Bitmap(imageSegmentation.cols, imageSegmentation.rows, pixelFormat);
                        BitmapData ^data = bmpImg->LockBits(Rectangle(0, 0, imageSegmentation.cols,
imageSegmentation.rows), ImageLockMode::WriteOnly, pixelFormat);

                        unsigned char *dstData = reinterpret_cast<unsigned char*>(data->Scan0.ToPointer());
                        unsigned char *srcData = imageSegmentation.data;

                        for (int row = 0; row < data->Height; ++row) {
                        memcpy(
                        reinterpret_cast<void*>(&dstData[row * data->Stride]),
                        reinterpret_cast<void*>(&srcData[row * imageSegmentation.step]),
                        imageSegmentation.cols * imageSegmentation.channels());
                        }

                        bmpImg->UnlockBits(data);

                        pictureBox1->Image = bmpImg;
```

```
*/

/*
int stride = imgSgm.size().width * imgSgm.channels();
int hDataCount = imgSgm.size().height;
Bitmap ^retImg;
IntPtr ptr(imgSgm.data);
if (stride % 4 != 0) {
uchar *dataPro = new uchar[((imgSgm.size().width * imgSgm.channels() + 3) & -4)];
uchar *data = imgSgm.ptr();
int curPosition = 0;
int curOffset = 0;
int offsetCounter = 0;

//itterate through all the bytes on the structure
for (int r = 0; r < hDataCount; r++) {
//fill the data
for (int c = 0; c < stride; c++) {
curPosition = (r * stride) + c;
dataPro[curPosition + curOffset] = data[curPosition];
}
//reset offset counter
offsetCounter = stride;
//fill the offset
do {
curOffset += 1;
dataPro[curPosition + curOffset] = 0;
offsetCounter += 1;
} while (offsetCounter % 4 != 0);
}
ptr = (System::IntPtr)dataPro;//set the data pointer to new/modified data array

//calc the stride to nearest number which is a multiply of 4
stride = (imgSgm.size().width * imgSgm.channels() + 3) & -4;

retImg = gcnew System::Drawing::Bitmap(imgSgm.size().width, imgSgm.size().height,
stride,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
ptr);
}
else {

//no need to add a padding or recalculate the stride
retImg = gcnew System::Drawing::Bitmap(imgSgm.size().width, imgSgm.size().height,
stride,
System::Drawing::Imaging::PixelFormat::Format24bppRgb,
ptr);
}
cli::array<unsigned char>^ imageData;
System::Drawing::Bitmap^ output;

// Create the byte array.
{
System::IO::MemoryStream^ ms = gcnew System::IO::MemoryStream();
retImg->Save(ms, System::Drawing::Imaging::ImageFormat::Png);
imageData = ms->ToArray();
delete ms;
}

// Convert back to bitmap
{
System::IO::MemoryStream^ ms = gcnew System::IO::MemoryStream(imageData);
output = (System::Drawing::Bitmap^)System::Drawing::Bitmap::FromStream(ms);
}
*/
```

```
                srcSgm.release();
                srcGray.release();
                srcFilter.release();
                src.release();
        }

        private: Void PictureECG_MouseDown(System::Object^ sender, System::Windows::Forms::MouseEventArgs^
e) {
                if (e->Button == System::Windows::Forms::MouseButtons::Left) {
                        region.X = e->X;
                        region.Y = e->Y;
                }
        }
        private: Void PictureECG_MouseUp(System::Object^ sender, System::Windows::Forms::MouseEventArgs^ e)
{
                if (e->Button == System::Windows::Forms::MouseButtons::Left && PictureECG->BackgroundImage
!= nullptr) {
                        region.Width = e->X - region.X;
                        region.Height = e->Y - region.Y;
                        if (region.Width > 0 && region.Height > 0)
                                ProcessingImage(ConvertDrawImageToCVMat(PictureECG->BackgroundImage,
region));
                }
        }
        private: Void PictureECG_MouseMove(System::Object^ sender, System::Windows::Forms::MouseEventArgs^
e) {
                if (e->Button == System::Windows::Forms::MouseButtons::Left && PictureECG->BackgroundImage
!= nullptr) {
                        region.Width = e->X - region.X;
                        region.Height = e->Y - region.Y;
                        Bitmap ^selectedImage = gcnew Bitmap(PictureECG->BackgroundImage->Width,
PictureECG->BackgroundImage->Height);
                        Graphics ^g = Graphics::FromImage(selectedImage);
                        System::Drawing::Color color = System::Drawing::Color::LightGreen; // цвет сетки
выделения
                        g->FillRectangle(gcnew HatchBrush(HatchStyle::Cross, color,
System::Drawing::Color::Empty), region);
                        g->DrawRectangle(gcnew System::Drawing::Pen(color), region);
                        PictureECG->Image = selectedImage;
                }
        }

        private: Void открытьToolStripMenuItem_Click(System::Object^ sender, System::EventArgs^ e) {
                if (OpenFileDialog->ShowDialog() == System::Windows::Forms::DialogResult::OK) {
                        PictureECG->BackgroundImage = Image::FromFile(OpenFileDialog->FileName);
                        PictureECG->Image = gcnew Bitmap(PictureECG->BackgroundImage->Width, PictureECG-
>BackgroundImage->Height);
                }
        }

        private: Void MainForm_Resize(System::Object^ sender, System::EventArgs^ e) {
                int w = this->ClientSize.Width - PanelPuctureECG->Location.X - 8;
                int h = this->ClientSize.Height - PanelPuctureECG->Location.Y - 12;
                PanelPuctureECG->ClientSize = System::Drawing::Size(w, h);

                w = this->ClientSize.Width - ChartECG->Location.X - 8;
                ChartECG->ClientSize = System::Drawing::Size(w, ChartECG->ClientSize.Height);
        }

        private: Bitmap ^Zoom(Image ^img, System::Drawing::Size size) {
                Bitmap ^btm = gcnew Bitmap(img, img->Width + (img->Width * size.Width / 100), img->Height +
(img->Height * size.Height / 100));
                Graphics ^g = Graphics::FromImage(btm);
                g->InterpolationMode = System::Drawing::Drawing2D::InterpolationMode::HighQualityBicubic;
```

```
                return btm;
        }
        private: System::Void выходToolStripMenuItem_Click(System::Object^  sender, System::EventArgs^  e) { this->Close(); }
};
}
```

Выпускная квалификационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«____» _____ _____ г.

Пенькова А.Е.