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

АВТОМАТИЗИРОВАННАЯ СИСТЕМА АНАЛИЗА ЭКГ С РАСШИРЕННЫМИ ВОЗМОЖНОСТЯМИ ПОИСКА ДИАГНОСТИЧЕСКИ ЗНАЧИМЫХ УЧАСТКОВ

Выпускная квалификационная работа
обучающегося по направлению подготовки 02.03.02 Фундаментальная информатика и информационные технологии
заочной формы обучения, группы 07001360
Берест Дмитрия Александровича

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

БЕЛГОРОД 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время компьютеры широко применяются в медицине. Без использования компьютерной техники не возможно обойтись на всех этапах медицинского обследования и диагностики, а также в профилактике заболеваний.

Предупреждение заболеваний на ранних стадиях развития является главной задачей медицины в наши дни. Поэтому разрабатывается различная диагностическая аппаратура, которая по своей сути представляет собой разновидности информационно-измерительных систем.

Одна из важнейших проблем современной медицины – рост числа сердечно-сосудистых заболеваний. Решение данной проблемы требует еще более широкого внедрения вычислительной техники в области исследования сердца. Здесь можно выделить два направления исследований:

- моделирование человеческого сердца;
- автоматизированная обработка данных кардиологических исследований.

В работе рассматриваются актуальные вопросы автоматизации обработки данных кардиологических исследований. Актуальность автоматизации обработки данных кардиологических исследований вызвана тем, что далеко не во всех случаях возможна ручная обработка таких данных. Это обусловлено тем, что, как правило, объемы данных значительны, а время на их обработку ограничено.

Также отметим, что в данном случае речь идет именно об автоматизированных, а не об автоматических системах, т.е. системы не могут полностью заменить врача. За постановку окончательного диагноза отвечает врач, а система должна рассматриваться как инструмент врача.

При автоматизированной обработке кардиологических данных можно выделить следующие наиболее важные преимущества:

- существенно сокращается время обработки данных;
- обработка данных осуществляется по одной схеме;
- результаты представляются в стандартном виде и др.

Целью выпускной квалификационной работы является создание автоматизированной системы анализа ЭКГ с расширенными возможностями поиска диагностически значимых участков.

Система основана на подходе к решению актуальной проблемы повышения эффективности работы врачей при оценки состояния сердца, который предложен в работах [1,2].

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить основы электрокардиографии.
2. Изучить структуру банк данных ЭКГ Physionet.
3. Изучить подход к построению системы компьютерного анализа ЭКГ с расширенными возможностями автоматизированного поиска характерных участков, предложенный в работах [1,2] и наработки по созданию данной системы, выполненные на кафедре математического и программного обеспечения информационных систем.
4. Выполнить постановку задачи, при этом сформулировать требования к функциональности разрабатываемой системы.
5. Разработать структурную схему программного обеспечения.
6. Обосновать выбор библиотеки обработки кардиосигналов.
7. Обосновать выбор инструментальных средств.
8. Разработать структуры данных.
9. Разработать модели программного обеспечения.
10. Выполнить программную реализацию системы.

11. Выполнить тестирование разработанного программного обеспечения на данных Physionet.

В первой главе произведен обзор метода электрокардиографического исследования, дана характеристика банка данных ЭКГ, используемого при тестировании разработанного программного обеспечения, выполнена постановка задачи.

Во второй главе разработана структура программного обеспечения, произведено обоснование выбора библиотеки обработки кардиосигналов, инструментальных и программных средств, выполнена разработка структур данных и ряда автоматных моделей, используемых при создании программного обеспечения.

В третьей главе рассмотрены вопросы программной реализации и тестирования системы.

Выпускная квалификационная работа содержит 59 страниц, 33 рисунка и 3 приложения на 11 страницах.

ГЛАВА 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1. Метод электрокардиографического исследования

При оценке состояния сердца и сердечно-сосудистой системы (ССС) пациента важную роль играет анализ электрокардиограммы (ЭКГ), которая представляет собой запись наблюдаемых на поверхности тела проекций объемных электрических процессов, происходящих в сердце в данный момент времени.

Первые электрокардиограммы были записаны Габриелем Липпманом с использованием ртутного электрометра. Эти записи лишь отдаленно напоминали современные ЭКГ. Опыты продолжил Виллем Эйнтховен, сконструировавший прибор, позволявший регистрировать ЭКГ. Он же ввел современные обозначения элементов ЭКГ (зубцов, интервалов) и описал некоторые нарушения в работе сердца. В 1924 году ему присудили Нобелевскую премию по медицине. Первая отечественная книга по электрокардиографии вышла под авторством русского физиолога А.Самойлова в 1909 г.

ЭКГ содержит информацию как о текущем состоянии ССС, так и о патологических изменениях в самом сердце. ЭКГ – является неинвазивным тестом, проведение которого позволяет получать ценную информацию о состоянии сердца. ЭКГ, как правило, отображается в графическом виде на дисплее или бумаге.

Во время работы сердца на поверхности тела возникают изменения разности потенциалов, которые записываются при помощи различных систем отведений ЭКГ. Каждое отведение регистрирует разность потенциалов, которая существует между двумя определенными точками электрического поля сердца, в которых установлены электроды. Таким образом, различные

электрокардиографические отведения отличаются друг от друга в первую очередь участками тела, от которых отводится разность потенциалов [3].

Существуют две распространенные скорости записи ЭКГ: 25 мм/с и 50 мм/с. ЭКГ аппарат обычно наносит скорость, при которой производилась запись ЭКГ на ленту. При записи 25 мм/с за 1 с. записывается 25 мм, т.е. 1 мм ленты (сторона маленькой клетки на ленте ЭКГ) соответствует 1/25=0.04 с., а 5 мм ленты (сторона большой клетки на ленте ЭКГ) соответствуют 0.04*5=0.2 с. При записи 50 мм/с за 1 с. записывается 50 мм, т.е. 1 мм ленты соответствует 1/50=0.02 с., а 5 мм ленты (сторона большой клетки на ленте ЭКГ) соответствуют 0.02*5=0.1 с.

Фрагменты ЭКГ записанные на скоростях 25 мм/с и 50 мм/с представлены на рис.1.1.

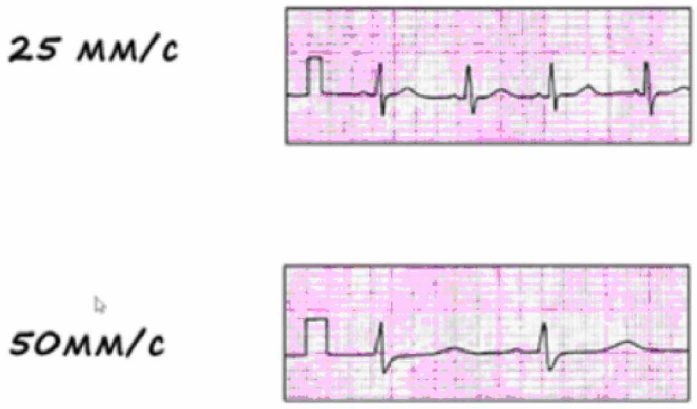


Рис. 1.1. Фрагменты ЭКГ при различной скорости записи

Обычно запись ЭКГ осуществляется в режиме 10 мм/мВ. ЭКГ аппарат обычно наносит выбранный вольтаж на ленту. Также перед началом записи ЭКГ аппарат генерирует прямоугольный импульс (калибровочный импульс). Данный импульс отображается на ленте. На рис.1.2 показан вольтаж и калибровочный импульс. Как видно из рисунка – высота импульса составляет 10 мм, т.е. при установленном режиме 10 мм/мВ, можно сделать вывод, что ЭКГ аппарат откалиброван правильно.

Рис.1.2. Вольтаж и калибровочный импульс

Под отведением ЭКГ понимают разность потенциалов между двумя электродами. Выделяют 12 отведений:

- Три стандартный отведения (I, II, III)
- Три усиленных отведения от конечностей (aVR, aVL, aVF)
- Шесть грудных отведений (V1, V2, V3, V4, V5, V6)

Стандартное отведение I фиксирует разность потенциалов от правой и левой руки. Стандартное отведение II фиксирует разность потенциалов от правой руки и левой ноги. Стандартное отведение III фиксирует разность потенциалов от левой руки и левой ноги. Правая нога в любом случае должна быть заземлена. Схематично стандартные отведения представлены на рис.1.3. Отведения I, II и III также можно представить в виде треугольника, названного треугольником Эйнтховена (рис.1.4).

Сначала ЭКГ состояла только из записи I, II, и III отведений. Потом стали использовать усиленные отведения от конечностей (рис.1.5). Эти отведения отличаются тем, что с одной стороны потенциал берется от конечности, а с другой стороны потенциал должен быть равен нулю. Ноль достигается объединением двух электродов от оставшихся конечностей через сопротивление.

Рис.1.3. Стандартные отведения

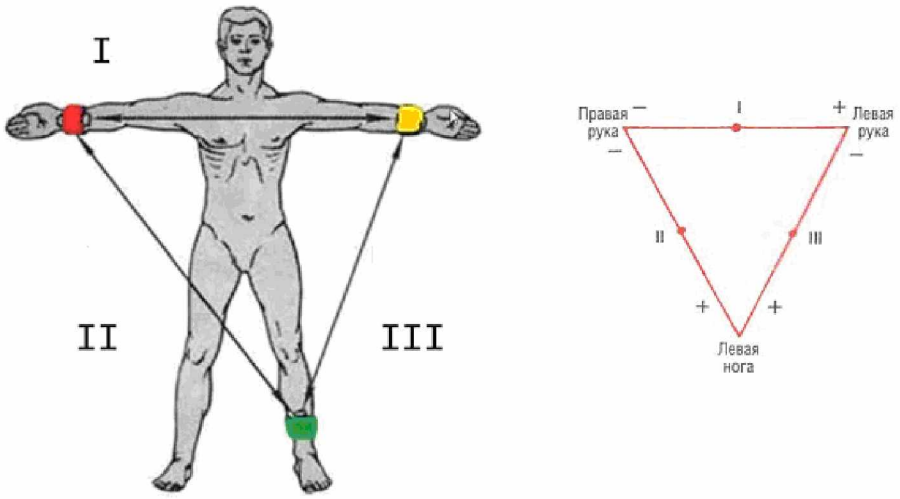


Рис.1.4. Стандартные отведения. Треугольник Эйнтховена

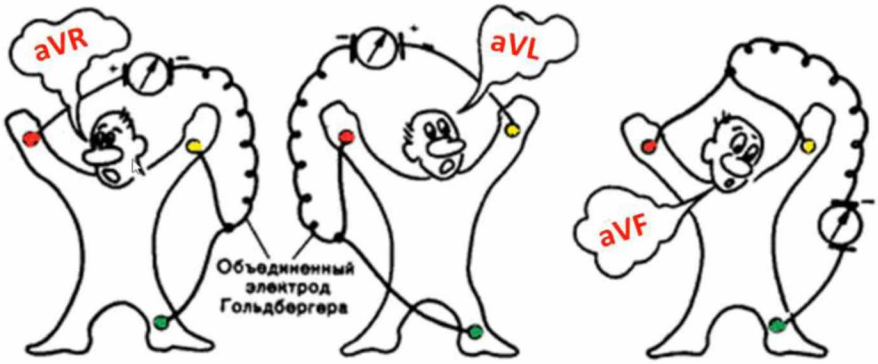


Рис.1.5. Усиленные отведения от конечностей

Также используются грудные отведения (рис.1.6). Они формируются аналогично усиленным отведениям. С одной стороны берется потенциал от

электрода, установленного на груди, а с другой стороны потенциал приближен к нулю, что достигается путем объединения трех электродов на конечностях, т.е. на правой руке, на левой руке и на левой ноге.

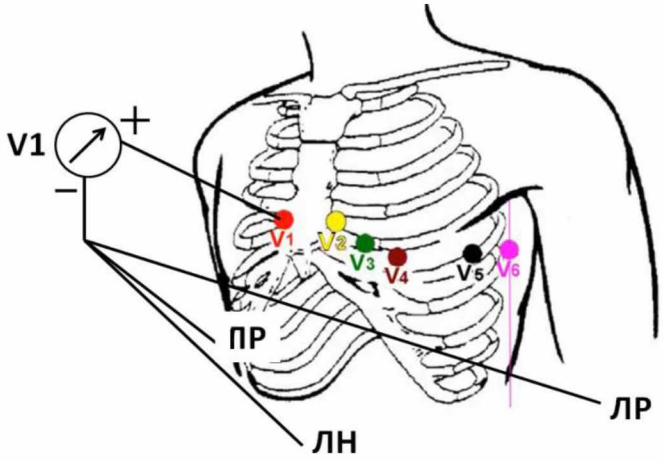


Рис. 1.6. Грудные отведения

Проведение ЭКГ сердца показано:

- перед любыми операциями;
- при оценке на пригодность к той или иной профессии (моряк, пилот и др.);
- при заболеваниях нервной системы, внутренних органов и других, если подозревается вовлечение сердечно-сосудистой системы в процесс патологии и возникновение осложнений;
- после инфекций, перенесенных человеком;
- в случае ухудшения состояния больного с заболеванием сердца;
- при появлении аритмии, одышки и болей в сердце.

Также регулярно проводить электрокардиографию сердца рекомендовано людям, которые находятся в зоне риска, а именно беременным, пожилым, курящим, больным гипертонией и другим.

Проанализировав данные электрокардиограммы, можно выявить врожденный порок сердца, увеличение размеров сердечной мышцы, различные аритмии, повреждение сердца при закупорке артерий, нарушение сердечного кровоснабжения, воспаление сердца, инфаркт миокарда.

Для расширения возможностей диагностики с помощью электрокардиографии также прибегают к длительной регистрации ЭКГ. Этот метод электрофизиологической инструментальной диагностики, называемый мониторированием ЭКГ по Холтеру, был предложен американским биофизиком Норманом Холтером. Диагностика осуществляется в течение 24 часов и более, иногда до 7 суток.

При помощи холтеровского мониторирования можно оценить деятельность сердца в различных ситуациях (нагрузки, сон и др.) Пациент должен записывать, что он делал в конкретное время. В ходе суточного мониторирования ЭКГ регистрация электрокардиограммы производится непрерывно на магнитную ленту или твердотельный диск. В дальнейшем полученная информация обрабатывается с помощью компьютера.

1.2.Основные элементы ЭКГ

Основными элементами, которые выделяют на ЭКГ, являются изолиния и зубцы. Зубцы, расположенные ниже изолинии, называют отрицательные. Зубцы, расположенные выше изолинии, называют положительными. На рис.1.7 представлен фрагмент ЭКГ с положительными зубцами – P,R,T и отрицательными зубцами – Q,S.

Также для диагностики важны интервалы, сегменты и комплексы. Наиболее важны два интервала: интервал P-Q и Q-T. Интервал P-Q включает часть ЭКГ от начала зубца P до начала зубца Q, т.е. данный интервал включает весь зубец P и кусочек изолинии от конца зубца P до начала зубца Q. Интервал Q-T включает часть ЭКГ от начала зубца Q до конца зубца T. Интервал Q-T состоит из зубца Q, зубца R, зубца S, изолинии от конца зубца S до начала зубца T и самого зубца T. Интервалы P-Q и Q-T представлены на рис.1.8. Между зубцами QRS нет изолинии. Эти зубцы называют комплексом QRS. Также выделяют два сегмента: сегмент P-Q и сегмент S-T. Сегмент P-Q

включает изолинию от конца зубца P до начала зубца Q. Сегмент S-T включает изолинию от конца зубца S до начала зубца T (см. рис.1.8).

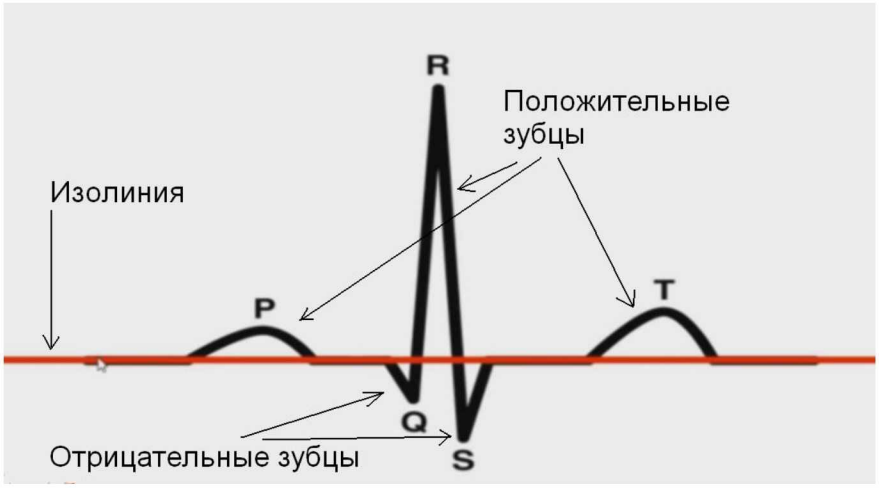


Рис.1.7. Изолиния и зубцы.

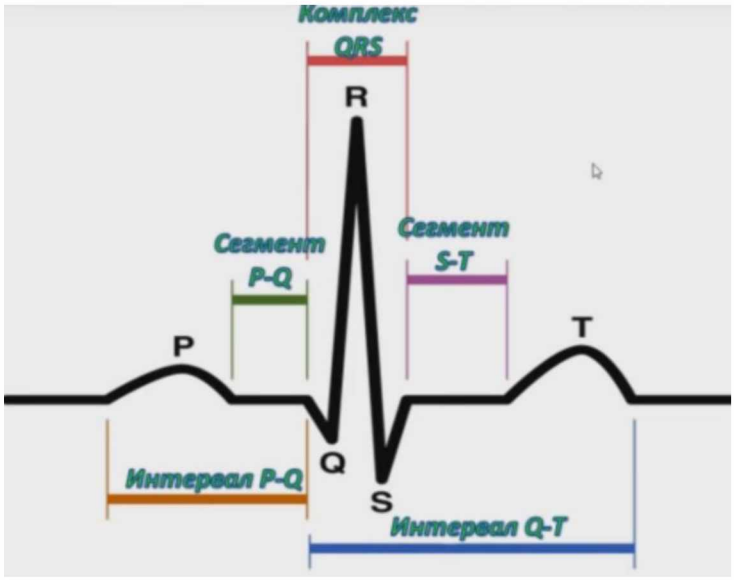


Рис.1.8. Интервалы, сегменты и комплексы.

1.3. Физиологический смысл основных элементов ЭКГ

Возбуждение сердца возникает в синусовом узле (сино-атриальном) узле (САУ) сино-атриальном узле (см. рис.1.9). Далее возбуждение передается на правое предсердие. Возбуждение можно изобразить схематично как некоторую волну (рис.1.9). От правого предсердия возбуждение передается на левое

предсердие. Возбуждение левого предсердия на рис.1.9 схематично изображено волной, аналогичной волне иллюстрирующей возбуждение правого предсердия, но выполненной с некоторой временной задержкой. Причем возбуждение левого предсердия начинается, когда возбуждения правого предсердия еще не завершено.

На кардиограмме процессы возбуждения правого и левого предсердий видны как зубец Р. Таким образом, зубец Р состоит из двух частей. Первая восходящая часть показывает процесс возбуждения правого предсердия, а вторая нисходящая часть показывает процесс возбуждения левого предсердия.

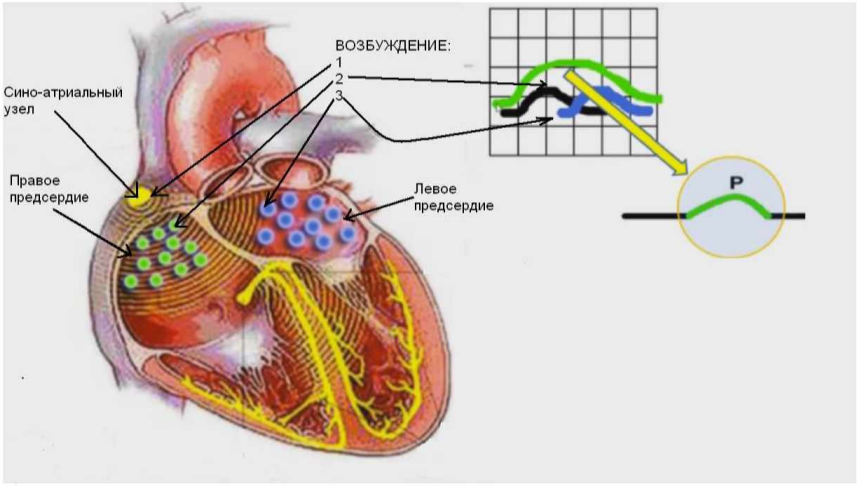


Рис.1.9. Зубец Р

После возбуждения предсердий происходит задержка возбуждения в атрио-вентрикулярном узле (АВ-узле). Величина задержки может достигать 0,08 с. При этом ничего не происходит, на ЭКГ видна изолиния (сегмент Р-Q). Таким образом, интервал Р-Q включает зубец Р, показывающий процесс возбуждения предсердий, и сегмент Р-Q, показывающий задержку возбуждения в АВ-узле (см.рис.1.10).

Комплекс QRS показывает возбуждение желудочков сердца. От АВ-узла возбуждение передается по пучку Гиса, расположенному в межжелудочковой перегородке. Пройдя некоторую часть пути, вектор возбуждение разворачивается в противоположную сторону и на ЭКГ регистрируется

отрицательный импульс Q, который показывает процесс возбуждения межжелудочковой перегородки. Отрицательный зубец Q возникает, т.к. вектор возбуждение направлен от регистрирующего электрода (см. рис.1.11).

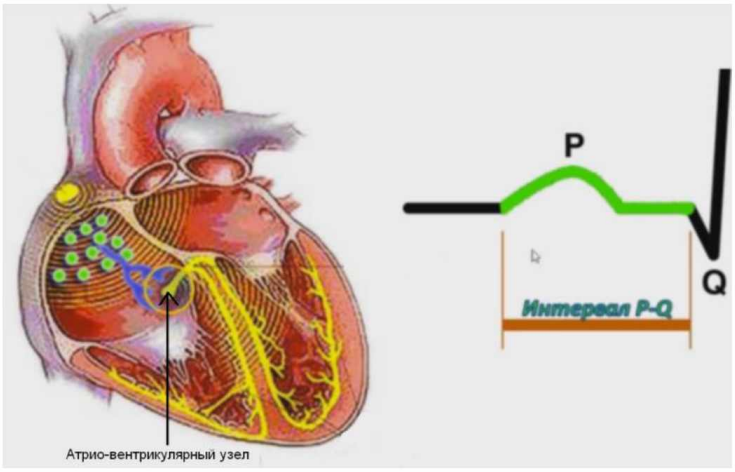


Рис.1.10. Интервал P-Q и сегмент P-Q

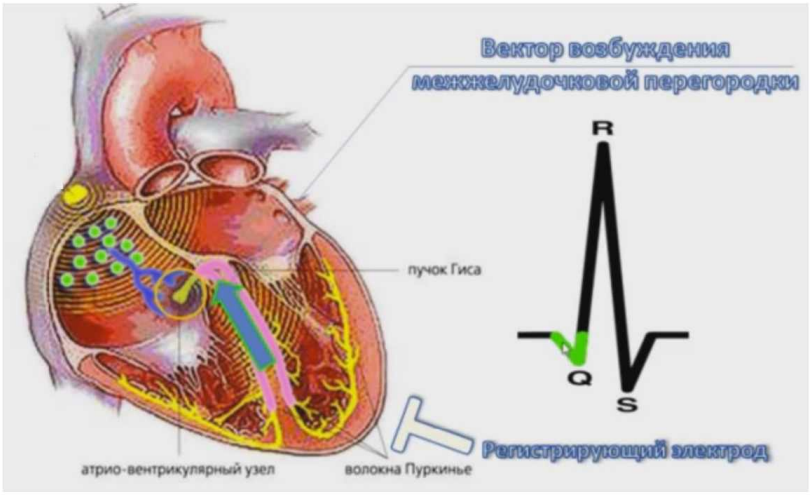


Рис.1.11. Зубец Q

Далее возбуждается верхушка сердца, и вектор возбуждения направлен к регистрирующему электроду. При этом регистрируется положительный зубец R (см. рис.1.12).

Затем возбуждается основание сердца. Вектор возбуждения в этом случае направлен от регистрирующего электрода и на ЭКГ фиксируется отрицательный зубец S (см. рис.1.13).

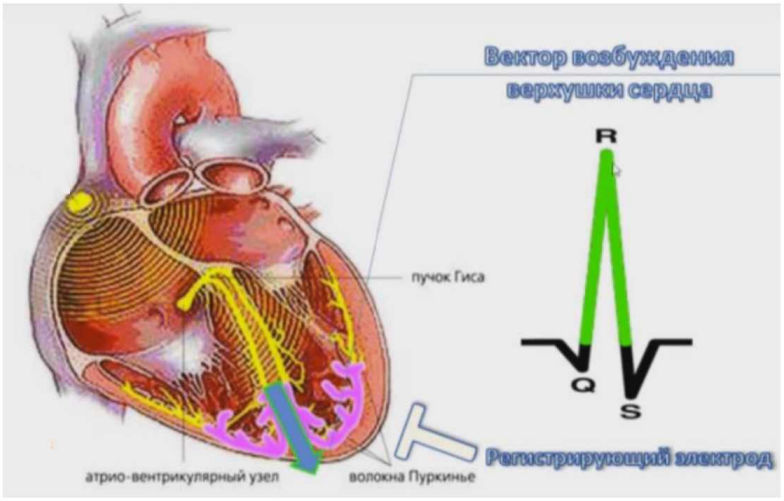


Рис.1.12. Зубец R

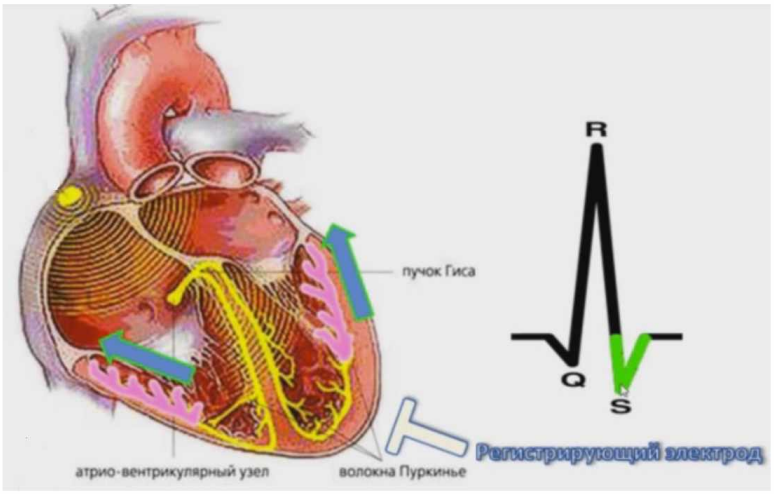


Рис.1.13. Зубец S

После того как возбуждение желудочков прошло нужно восстановить желудочки до исходного уровня. Это плавный переход. Когда еще идет процесс возбуждения, но уже начался процесс восстановления, оба процесса накладываются друг на друга и некоторое время оба процесса в сумме дают ноль, на ЭКГ это отображается в виде сегмента S-T. Зубец Т возникает, когда процесс восстановления преобладает над процессом возбуждения (см. рис.1.14).

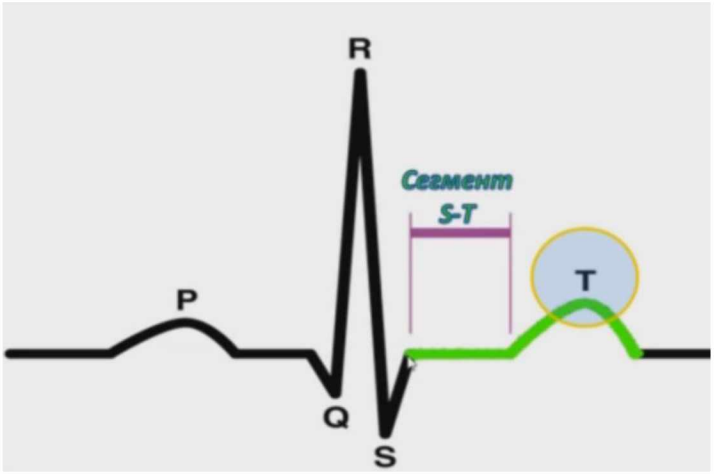


Рис.1.14. Сегмент S-T и зубец T

1.4. Банк данных ЭКГ PhysioNet

Ресурс для научных исследований и образования PhysioNet предлагает бесплатный доступ через Интернет к большому объему цифровых записей физиологических сигналов и программному обеспечению с открытым исходным кодом для обработки таких данных [4].

PhysioNet был создан в качестве механизма для открытого и свободного распространения, обмена записанными биомедицинскими сигналами и программным обеспечением с открытым исходным кодом для их анализа. Все данные, включенные в PhysioBank, и все программное обеспечение, включенное в PhysioToolkit, тщательно анализируются.

PhysioNet состоит из трех компонентов:

- PhysioBank - архив цифровых записей физиологических сигналов, временных рядов и связанных с ними данных, которые используются в биомедицинских исследованиях.
- PhysioToolkit - библиотека программного обеспечения для физиологической обработки сигналов, анализа и выявления физиологически значимых событий.
- PhysioNetWorks - виртуальная лаборатория, в которой можно вместе с командой PhysioNet и коллегами из любой точки мира работать над созданием,

оценкой, улучшением, документированием, а также подготовкой новых данных и рабочего программного обеспечения для публикации на PhysioNet.

Детальная структура ресурса PhysioNet представлена в приложении 1.

1.5. Постановка задачи

Компьютерный анализ ЭКГ предусматривает реализацию следующих этапов:

1. Подготовка входных данных.

2. Распознавание особых точек и участков ЭКГ (отдельных зубцов, комплексов QRS, QT и др.).

3. Автоматизированный анализ результатов, полученных на шаге 2.

4. Постановка диагноза.

На первом этапе осуществляется сбор кардиологической информации и внесение ее в компьютер - рабочее место (РМ) врача. Для реализации этого этапа может использоваться широкий набор средств: от электрокардиографа, сопряженного с РМ врача до сложных сетевых систем, позволяющих осуществлять сбор информации с электрокардиографов различных типов по проводным и беспроводным каналам связи. Системы сбора кардиографической информации, как правило, могут быть сконфигурированы для требуемого уровня сложности.

Второй этап обычно выполняется в автоматическом режиме с помощью специальных алгоритмов цифровой обработки сигналов. Существует целый ряд подобных алгоритмов.

Третий этап, как правило, выполняется с помощью алгоритмов, реализующих различные методики анализа ЭКГ.

Заключительный этап выполняется врачом. Компьютер в данном случае используется для автоматизации поиска, внесения и хранения информации.

Использование врачом компьютерного анализа ЭКГ для постановки диагноза может существенно снизить время решения этой задачи, а также исключить грубые врачебные ошибки. При этом одним из наиболее перспективных путей повышения эффективности работы врача является улучшение пользовательского интерфейса РМ врача. Можно выделить следующие подходы к решению этой задачи:

1. Следование принципам юзабилити.

2. Улучшение качества алгоритмов автоматического поиска особых точек ЭКГ.

3. Реализация средств автоматизированного поиска характерных участков ЭКГ.

В данной работе реализуется система компьютерного анализа ЭКГ, в которой используется автоматическое распознавание особых точек ЭКГ (аннотирование ЭКГ) и реализованы расширенные возможности автоматизированного поиска характерных участков (линий) ЭКГ. Каждой особой точке ЭКГ (аннотированной точке) присваивается специальный код (код аннотации) [1].

Каждая линия соответствует участку ЭКГ, начинающегося в аннотированной точке с заданным кодом, включая эту точку, и заканчивающегося в следующей аннотированной точке с заданным кодом, не включая эту точку [1].

Система должна обладать следующей функциональностью:

1. Ввод данных ЭКГ из файлов в формате банка PhysioBank

2. Просмотр сигнала ЭКГ на графике с возможностью выбора и просмотра отдельных участков (окон) ЭКГ.

3. Запоминание параметров выбранных окон и передвижение по ним.

4. Автоматическое аннотирование ЭКГ.

5. Перемещение по аннотированным точкам.

6. Перемещение по зубцам R.

7. Выбор начального кода аннотации.

8. Автоматический поиск линий ЭКГ.

9. Выбор линии ЭКГ.

10. Перемещение по выбранным линиям ЭКГ.

Пункты 4-10 расширяют возможности поиска диагностически значимых участков ЭКГ по сравнению с традиционными системами. И если реализация функций аналогичных функциям 4-6 встречается в подобных системах, то функции 7-10 являются новыми [1,2]. Исходя из сказанного, в дальнейшем реализуемую систему будем называть автоматизированной системы анализа ЭКГ с расширенными возможностями поиска диагностически значимых участков.

Также отметим, что на кафедре математического и программного обеспечения информационных систем реализован прототип системы, в котором реализованы функции 1-6. Данный прототип выбран за основу системы, разрабатываемой в данной работе. Некоторые из функций 1-6 модифицируются, функции 7-10 полностью разрабатываются в рамках данной выпускной квалификационной работы.

ГЛАВА 2. ПРОЕКТНАЯ ЧАСТЬ

2.1. Разработка структурной схемы программного обеспечения

Разработка структурной схемы программного обеспечения является одним из наиболее важных этапов в процессе его разработки по следующим причинам:

- неправильный выбор структуры программного обеспечения ведет к риску срыва всего проекта в будущем;
- хорошо продуманная структура программного обеспечения позволяет легко модифицировать программный продукт, если произойдут изменения требований к нему.

Взаимодействие информационных процессов в системе компьютерного анализа ЭКГ представим с помощью диаграмм IDEF3. На данных диаграммах показывается последовательность действий или процессов анализируемой системы. Диаграммы IDEF3 отображают действие в виде прямоугольника. Связи между действиями изображаются с помощью стрелок. Стрелка может начинаться или заканчиваться на любой стороне блока. В модели IDEF3 определены три типа связей [5]. В нашем случае используется связь «Временное предшествование». Смысл данной связи в том, что исходное действие должно завершиться прежде, чем конечное действие сможет начаться.

Также в моделях IDEF3 используются соединения. Соединения разбивают или соединяют внутренние потоки и используются для описания ветвления процесса. Различают следующие виды соединений:

1. Разворачивающие соединения. Данные соединения используются для разбиения потока. Завершение одного действия вызывает начало выполнения нескольких других действий.

2. Сворачивающие соединения. Данные соединения используются для объединения потоков. Завершение одного или нескольких действий вызывает начало выполнения только одного другого действия.

Соединения также подразделяются на асинхронные и синхронные. В нашем случае будут использоваться только асинхронные соединения. В модели IDEF3 определены три типа асинхронных соединений. Обозначения данных соединений представлены в таблице 2.1.

Таблица 2.1

Типы соединений в модели IDEF3

Графическое обозначение	*Название*	*Вид*	*Правила инициации*
	Соединение «И»	Разворачивающее	Каждое конечное действие обязательно инициируется
		Сворачивающее	Каждое исходное действие обязательно должно завершиться
	Соединение «ИЛИ»	Разворачивающее	Одно (или более) конечное действие инициируется
		Сворачивающее	Одно (или более) исходное действие должно завершиться
	Соединение «Исключающее ИЛИ»	Разворачивающее	Одно и только одно конечное действие инициируется
		Сворачивающее	Одно и только одно исходное действие должно завершиться

Диаграмма верхнего уровня, иллюстрирующая взаимодействие информационных процессов в системы компьютерного анализа ЭКГ, представлена на рис.2.1.

Вначале требуется выбрать файл и канал ЭКГ. После такого выбора данные ЭКГ загружаются в память компьютера и отображаются на экране в графическом виде.

Также отображаются следующая информация: имя загруженного файла, число каналов, номер выбранного канала, время снятия ЭКГ, число отсчетов, разрядность, частота и амплитуда сигнала ЭКГ, и, возможно, другая информация, представленная в файлах ЭКГ в формате Physionet. Более

детально отображаемая информация будет уточнена на этапе разработки интерфейса системы.

После считывания данных ЭКГ выполняется автоматическое аннотирование. Также выделяются точки R-R или QS-QS (точки RRS), представляющие собой интервал между сокращениями желудочков.

После автоматического аннотирования пользователь может осуществлять просмотр ЭКГ. Детализация данного действия будет рассмотрена ниже. Закончив просмотр, врач формирует диагноз (медицинское заключение). Автоматизация этого процесса в рамках данной выпускной квалификационной работы не предусмотрена. Это связано с тем, что для выполнения подобных действий, существует много различного программного обеспечения. Как правило, такое программное обеспечение относится к классу медицинских информационных систем. Обычно такие системы разрабатываются в рамках отдельного медицинского учреждения или региона.

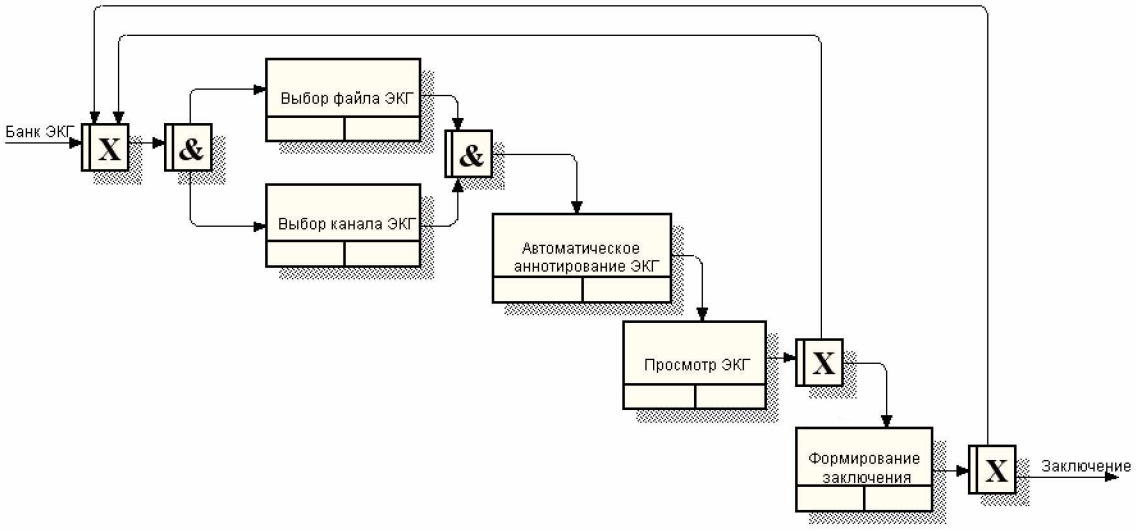


Рис.2.1. Диаграмма IDEF3, иллюстрирующая взаимодействие информационных процессов в системы компьютерного анализа ЭКГ

Детализация функции просмотра ЭКГ иллюстрирует IDEF3 диаграмма, представленная на рис.2.2. В любой момент работы имеется возможность отобразить произвольный участок (окно) ЭКГ. Для этого с помощью курсоров

необходимо выбрать границы окна. Выделенные окна можно запоминать. Также возможно осуществлять передвижение по последовательности запомненных окон, по особым точкам и линиям ЭКГ.

Перемещение по особым точкам ЭКГ осуществляется в двух режимах: перемещение по аннотированным точкам ЭКГ и перемещение по точкам RRS. При этом в окне ЭКГ текущая точка помечается соответствующим курсором. На экран выводится число особых точек в выбранном окне ЭКГ. Для текущей точки также выводится номер отсчета сигнала ЭКГ, код аннотации и сама аннотация.

Для удобства поиска специфических участков ЭКГ можно выбрать начальный код аннотации. После такого выбора осуществляется автоматический поиск линий. Линия может встречаться в ЭКГ несколько раз. Если выбрать линию, то перемещение по начальным точкам этой линии осуществляется аналогично перемещению по особым точкам ЭКГ. В окне ЭКГ начальная точка выбранной линии также помечается соответствующим курсором.

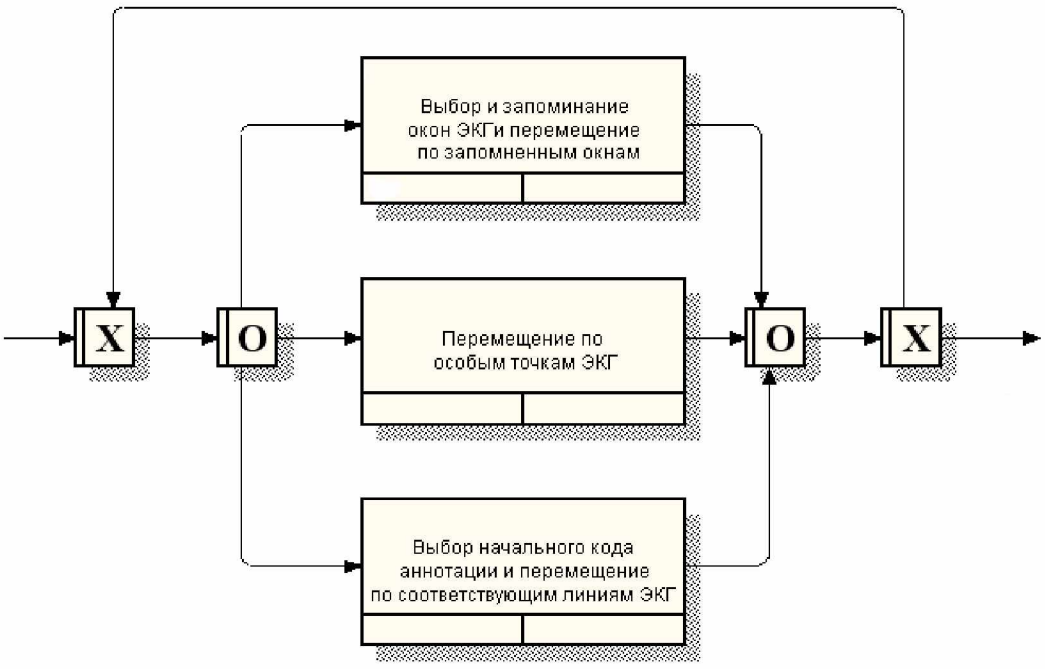


Рис.2.2. Диаграмма IDEF3, детализирующая блок «Просмотр ЭКГ»

Средства удобного поиска специфических участков ЭКГ, а также перемещения по особым точкам являются наиболее важными в разрабатываемой системе. Именно наличие этих средств позволяет врачу быстро найти диагностически значимые участки ЭКГ и тем самым повысить эффективность работы врача.

2.2. Обоснование выбора библиотеки обработки кардиосигналов

Как было отмечено в разд.1.5, компьютерный анализ ЭКГ предусматривает реализацию следующих этапов:

1. Подготовка входных данных.

2. Распознавание особых точек и участков ЭКГ (отдельных зубцов, комплексов QRS, QT и др.).

3. Автоматизированный анализ результатов, полученных на шаге 2.

4. Постановка диагноза.

Отметим, что в разрабатываемой системе первый этап сводится к разработке средств считывания данных ЭКГ из файлов в формате Physionet. При дальнейшем развитии системы могут добавляться другие способы получения данных ЭКГ.

Четвертый этап, как было отмечено ранее, в рамках данной выпускной квалификационной работы не автоматизируется.

Основное внимание в работе уделяется автоматизации второго и третьего этапов. При этом третий этап сводится к разработке средств расширенного поиска диагностически значимых участков ЭКГ, что повышает эффективность работы врача при постановке диагноза. Но, даже самые хорошие средства такого поиска не принесут желаемого результата, если распознавание участков ЭКГ, которые имеют диагностическое значение, выполняемое на втором этапе, будет плохо реализовано.

Рассмотрим существующие средства автоматизации второго этапа более детально. Существует много готовых алгоритмов распознавание участков ЭКГ. В работе [6] приведены результаты тестирования 12 таких алгоритмов. Сравнительный анализ алгоритмов был проведен по следующим параметрам: методология, устойчивость к шумам, точность выявления особых точек ЭКГ, возможность настройки, время обработки и др. В этой работе был сделан вывод о том, что алгоритм, представленный в [7], обладает наименьшим временем обработки сигнала и хорошо определяет особые точки ЭКГ. Результаты сравнительного анализа, выполненного в работе [6], представлены на рис.2.3.

	Алгоритм 1	Алгоритм 2	Алгоритм 3	Алгоритм 4	Алгоритм 5	Алгоритм 6	Алгоритм 7	Алгоритм 8	Алгоритм 9	Алгоритм 10	Алгоритм 11	Алгоритм 12
Методология	Низко-частотная и высоко-частотная фильтрация алгоритмы непрерыв-ного и быстрого вейвлет-преобразования	Основан на Pan&Tompkins	-	Низкочастотная и высокочастотная фильтрация, адаптивная пороговая фильтрация	Улучшенный алгоритм Pan&Tompkins	Вейвлет-преобразования	Интерполяция сигнала для удаления дрифта изолинии, фильтрация для удаления шума, пороговая фильтрация для детекции пиков.	Фильтрация шума и дрифта, детекция пиков с помощью пороговой фильтрации	Фильтрация шума и дрифта изолинии с помощью низкочастотной и высокочастотной фильтрации, пороговая фильтрация	Фильтрация шума и дрифта изолинии с помощью низкочастотной и высокочастотной фильтрации, пороговая фильтрация	Фильтрация шума и дрифта изолинии с помощью низкочастотной и высокочастотной фильтрации, пороговая фильтрация	Основан на теории огибающий (Envelope)
Устойчи-вость к шумам	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту	устойчив к шумам и дрифту
Точность	высокая	высокая	высокая	высокая	высокая	высокая	-	-	-	-	-	-
Адаптив-ность	без подстройки	без подстройки	без подстройки	без подстройки	без подстройки	без подстройки	требуется подстройка	без подстройки	без подстройки	без подстройки	без подстройки	без подстройки
Настройка	возможна	отсутствует	возможна	возможна	возможна	возможна	возможна	возможна	возможна	возможна	возможна	возможна
Диапазон частот	>128Гц	-	-	120 ГЦ	200Гц и 250 Гц	-	-	-	-	-	-	-
Время обработки	9 мс	0,5 с	0,1 с	0,11 с	-	0,25 с	0,19 с	0,17 с	0,13 с	0,009 с	0,02 с	0,007 с
Проблемы	-	не правильно определяет QRS,P,T	не правильно определяет начало QRS	1)необходимо нормиро-вать сигнал 2)не определяет пик R	Не удалось запустить исходный код	1) не все характеристики определя-ются точно	не стабилен в детекции пиков Q,S	не выполняет детекцию пиков Q	определяет только положение пика R	1) алгоритм определяет лишь пики S,R 2) иногда генерирует ложные детекции	1) опреде-ляет только пики R 2)не опреде-ляет пики в начале и конце сигнала	1) алгоритм определяет только пики R 2)в начале и конце могут быть ложные пики

Рис. 2.3. Результаты анализа алгоритмов компьютерного анализа ЭКГ

Основываясь на анализе, выполненном в работе [6], и собственных экспериментах, было принято решение использовать алгоритм, представленный в [7]. Данный алгоритм реализован в виде библиотеки ECG Annotation C++ Library, имеющей лицензию GNU GPL, версии 3, с помощью которой также можно вводить данные из файла ЭКГ в формате Physionet. Таким образом, с помощью данной библиотеки, можно решить ряд основных задач, возникающих на первом и втором этапах компьютерного анализа ЭКГ (см. выше).

2.3. Обоснование выбора инструментальных средств

Для разработки автоматизированной системы анализа ЭКГ с расширенными возможностями поиска диагностически значимых участков была выбрана графическая среда программирования LabVIEW (Laboratory Virtual Instrument Engineering Workbench). Разработчиком LabVIEW является американская компания National Instruments [8-11].

С помощью LabVIEW можно разрабатывать прикладные программные продукты позволяющие собирать, обрабатывать и отображать информацию и результаты расчетов, взаимодействовать с измерительной и управляющей аппаратурой, а также моделировать не только отдельные объекты, но и автоматизированные системы в целом.

В отличие от текстовых языков программирования, таких как C, Pascal, Java и других, где программы составляются в виде строк текста, в LabVIEW программы составляются не в виде строк кода, а в виде графических диаграмм.

LabVIEW – это открытая система программирования, которая поддерживает различные современные программные интерфейсы (Win32 DLL, COM.NET и др.). Также в состав LabVIEW входят библиотеки управления различными аппаратными интерфейсами и средствами, например PCI. Кроме того, разработанное с использованием LabVIEW программное обеспечение может быть дополнено фрагментами, которые разработаны при помощи традиционных языков программирования, таких как C/C++, Pascal, Basic, FORTRAN, а также модули, которые были разработаны в LabVIEW, можно использовать в проектах, которые создаются в других системах программирования.

Таким образом, с помощью LabVIEW можно создавать фактически любые приложения, которые будут взаимодействовать с любыми видами аппаратных средств, поддерживаемых операционной системой компьютера.

Безусловно, достоинством LabVIEW является и то, что разработчику и пользователю доступны функционально одинаковые системы программирования для различных операционных систем, таких как Microsoft Windows, Linux, MacOS. Таким образом, программа, которая была разработана для Windows, будет практически без изменений работать на компьютере с MacOS.

Также, система программирования LabVIEW включает в себя встроенный механизм отладки приложений. В процессе отладки можно назначать точки останова программы, выполнять программу по шагам, визуализировать процесс исполнения программы и контролировать любые данные в любом месте программы.

Кроме того, система LabVIEW позволяет защитить программы от несанкционированного изменения или просмотра их исходного кода. Для этого следует воспользоваться паролями для доступа к приложениям или полностью удалить исходный код из работающего приложения.

Таким образом, можно сделать вывод, что LabVIEW представляет собой высокоэффективную среду графического программирования, предназначенную для реализации функционально гибких и масштабируемых приложений.

Как было отмечено выше, при создании системы будет использоваться библиотека, представленная в [7]. Данная библиотека написана на языке C++ и распространяется в виде исходного кода. Поэтому еще одним инструментальным средством, использующимся при создании системы, будет среда, поддерживающая программирование на C++ [12].

Связь программы на LabVIEW и библиотеки планируется осуществлять через написание dll, которые будем создавать на основе кода, представленного в [7].

Было принято решение использовать Microsoft Visual Studio 2008 [13]. Выбор версии Microsoft Visual Studio обусловлен только тем, что при написании выпускной квалификационной работы была доступна лицензионная

версия Microsoft Visual Studio 2008. Можно было использовать и другие инструментальные средства, поддерживающие создание dll на основе кода C++.

2.4. Разработка структур данных и автоматных моделей, используемых при создании программного обеспечения

При разработке функций, реализующих расширенные возможности автоматизированного поиска характерных участков (линий) ЭКГ использовался автоматный подход к проектированию программного обеспечения.

При применении данного подхода программа или ее фрагмент представляется с помощью автоматных моделей. В зависимости от конкретной задачи могут использоваться как конечные автоматы, так и автоматы более сложной структуры.

Следует отметить, что усложнение автоматных моделей приводит к сложностям понимания логики работы программной системы и тем самым теряется основное преимущество автоматного подхода к проектированию программного обеспечения. Однако в некоторых ситуациях, например, при проектировании синтаксических распознавателей контекстно-свободных языков, усложнение автоматных моделей необходимо. В этом случае следует использовать автоматы с магазинной памятью, т.к. конечные автоматы не могут распознавать контекстно-свободные языки. В нашем случае достаточно использовать конечные автоматы.

Конечный автомат является хорошим средством формальной структуризации приложения. Вместо того чтобы использовать все переменные приложения в качестве расширенного определения его состояния, конечный автомат создает единственную переменную, в которой хранится информация о состоянии приложения. Такой переменной обычно является тем или иным образом заданный список текущих состояний автомата.

Особенность рассматриваемого подхода состоит в том, что при его использовании автоматы задаются графами переходов, а программирование заключается в написании программы, моделирующий построенный на этапе проектирования конечный автомат.

При построении моделирующей программы можно использовать компиляционный и интерпретационный подходы.

Рассмотрим абстрактный конечный автомат представленный диаграммой на рис.2.4. Здесь и далее используются диаграммы состояний UML [14-15].

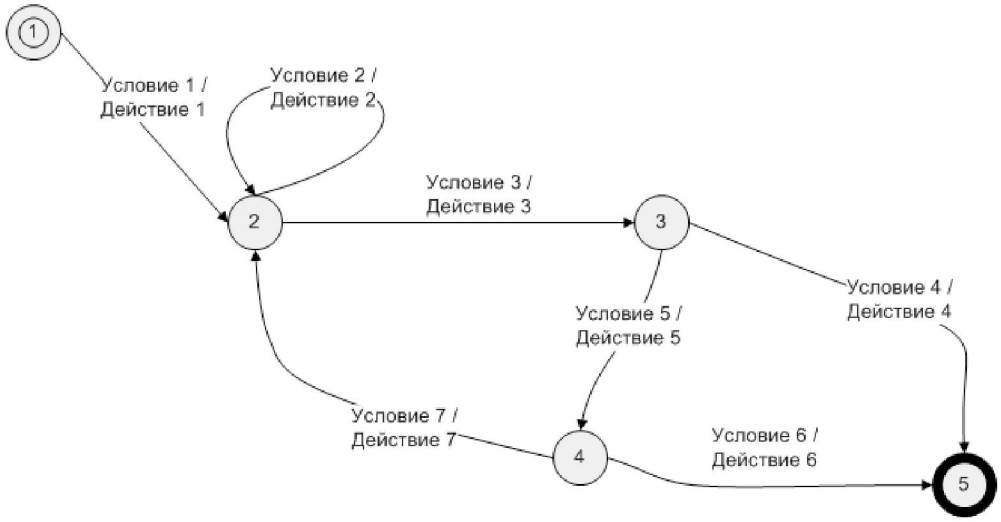


Рис. 2.4. Пример автомата.

Компиляционный подход заключается в том, что каждому конечному автомату соответствует своя программа. При использовании компиляционного подхода для реализации данного автомата нужно использовать программу, код которой содержит фрагмент аналогичный следующему:

Листинг 2.1. Фрагмент программы.

```
_m1: if (Условие 1) {Действие 1; goto 2;}
goto _m1;
_m2: if (Условие 2) {Действие 2; goto 2;}
if (Условие 3) {Действие 3; goto 3;}
goto _m2;
```

Листинг 2.1. Фрагмент программы. Продолжение.

```
_m3: if (Условие 4) {Действие 4; goto 5;}
if (Условие 5) {Действие 5; goto 4;}
goto _m3;
_m4: if (Условие 6) {Действие 6; goto 5;}
if (Условие 7) {Действие 7; goto 2;}
goto _m4;
_m5: return e;
```

При использовании интерпретационного подхода для реализации автоматов нужно разработать способ их кодирования в памяти ЭВМ и программу, интерпретирующую эти данные. Такая программа выбирает данные, относящиеся к текущему состоянию, вычисляет условия, приписанные переходам из этого состояния, выполняет соответствующий переход и действие. Таким образом, при реализации нескольких автоматов используется одна программа, изменяются только данные.

В нашем случае более оправдано применение компиляционного подхода для реализации автоматов. Это обусловлено тем, что требуется реализовать небольшое количество автоматов, при этом используется компилятор C++. Также отметим, что при использовании компиляционного подхода быстродействие, как правило, выше, что в нашем случае имеет большое значение.

Рассмотрим структуры данных, которые используются в системе. Схема данных представлена на рис.2.5.

Исходные данные программы (данные ЭКГ) представляются собой массив вещественных чисел D размера lenD. Элемнт массива D[i] содержит оцифрованное значение сигнала ЭКГ в i-ый момент времени (отсчет сигнала ЭКГ).

Аннотация ЭКГ представляют собой язык $L = \varpi^*$, где ϖ – алфавит, состоящий из символов, которые могут быть присвоены особым точкам ЭКГ при выполнении автоматического аннотирования. Язык L относится к классу регулярных языков.

Аннотация ЭКГ сохраняется в виде двух массивов AT и AS размера lenA. Элемент AT[j] содержит код аннотации присвоенный j-ой особой точке ЭКГ. Элемент AS[j] содержит номер отсчета этой особой точки.

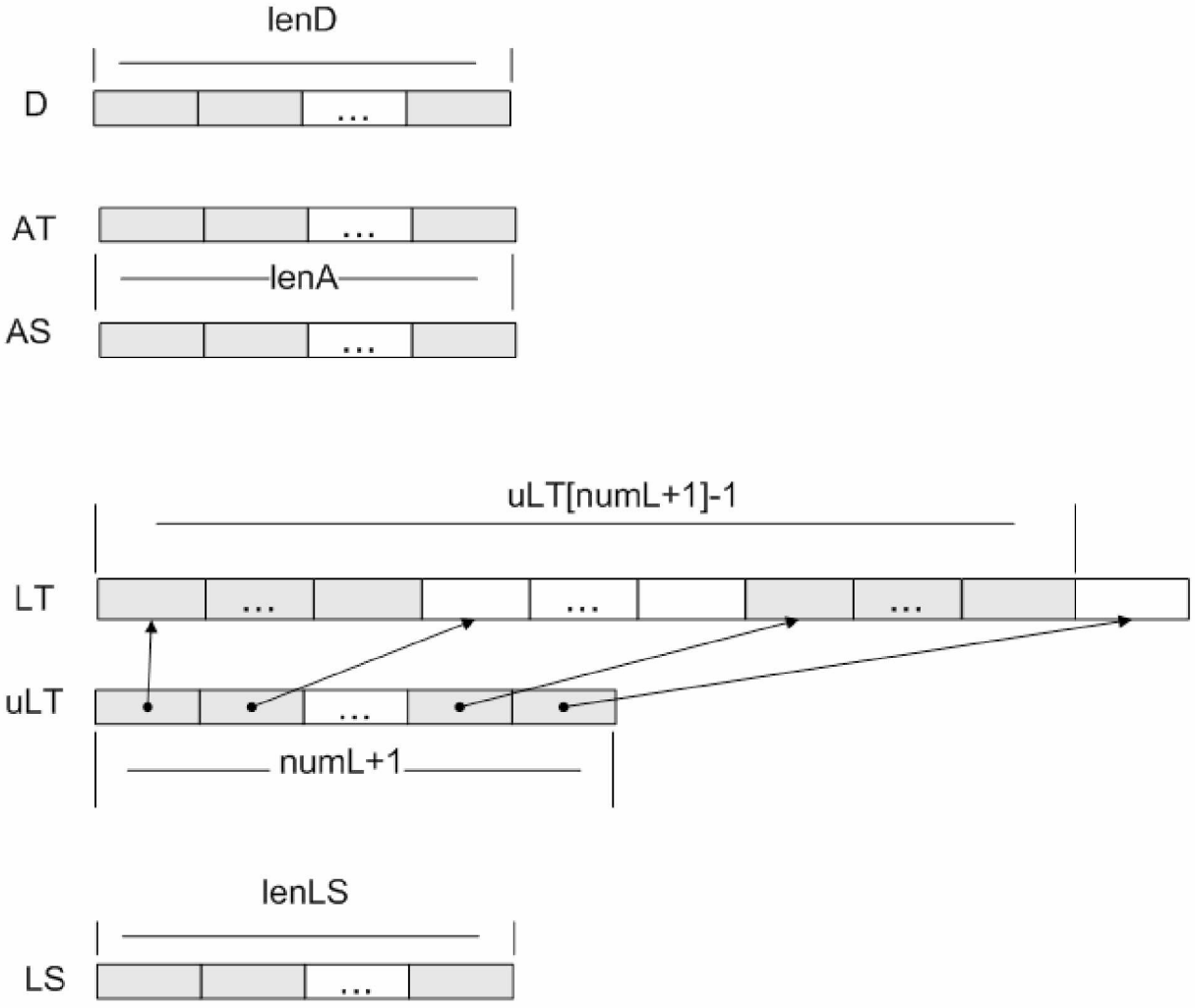


Рис.2.5. Структуры данных

Напомним, что каждая линия состоит из участка ЭКГ начинающегося в аннотированной точке с заданным кодом, включая эту точку, и

заканчивающегося в следующей аннотированной точке с заданным кодом, не включая эту точку.

Линии ЭКГ сохраняются в структуре, состоящей из двух массивов LT и uLT (рис.2.5). На начало i-ой линии указывает i-ый элемент массива uLT размера numL+1, где numL – число линий. Таким образом, i-я линия ЭКГ состоит из аннотированных точек со следующими кодами: LT[uLT[i]],…,LT[uLT[i+1]-1], а размер массива LT равен uLT [numL+1]-1.

Список начальных отсчетов выбранной линии хранится в массиве LS размера lenLS (рис.2.5).

Формирование массивов LT и uLT осуществляет конечный автомат, представленный на рис.2.6.

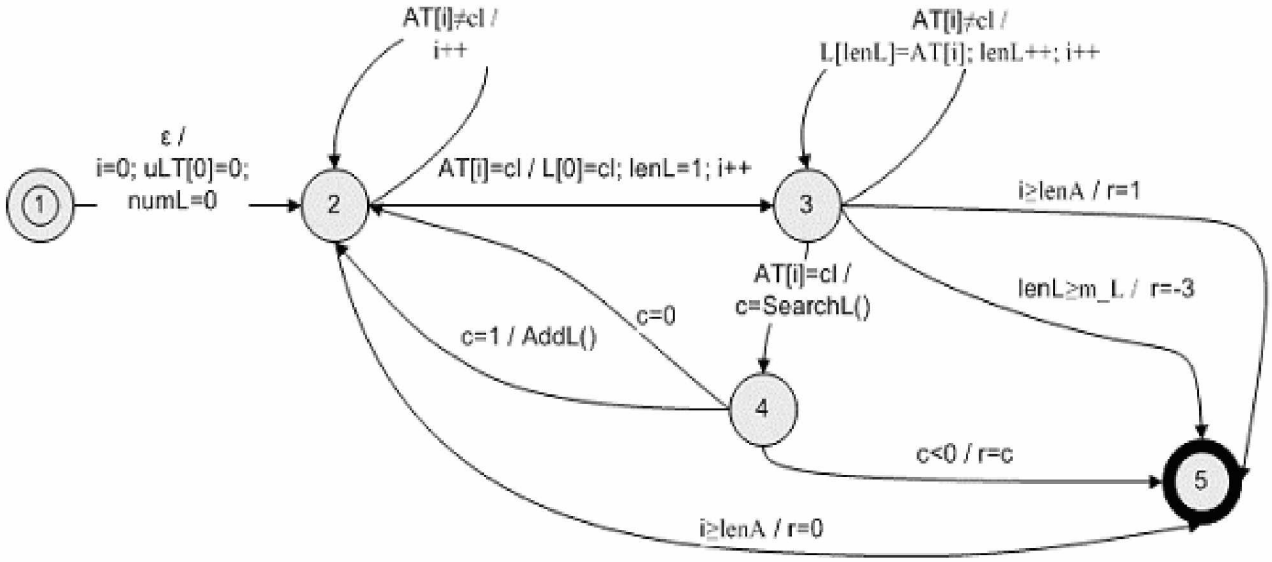


Рис.2.6. Диаграмма состояний конечного автомата формирующего структуру для хранения линий ЭКГ

Код аннотации начальной точки линии хранится в переменной cl. Текущая линия сохраняется в массиве L длины lenL. Также в автомате используются функции SearchL() и AddL().

Функция SearchL() осуществляет поиск линии L в структуре LT, uLT. Данная функция возвращает:

- 0 - линия аннотации не уникальна;

- 1 - линия аннотации уникальна и может быть добавлена в структуру LT, uLT;
- -1 - линия аннотации уникальна, но для ее добавления в структуру LT, uLT нет места в массиве LT;
- -2 - линия аннотации уникальна, но для ее добавления в структуру LT, uLT нет места в массиве uLT.

Алгоритм работы функции SearchL() представлен диаграммой деятельности [14,15] на рис. 2.7.

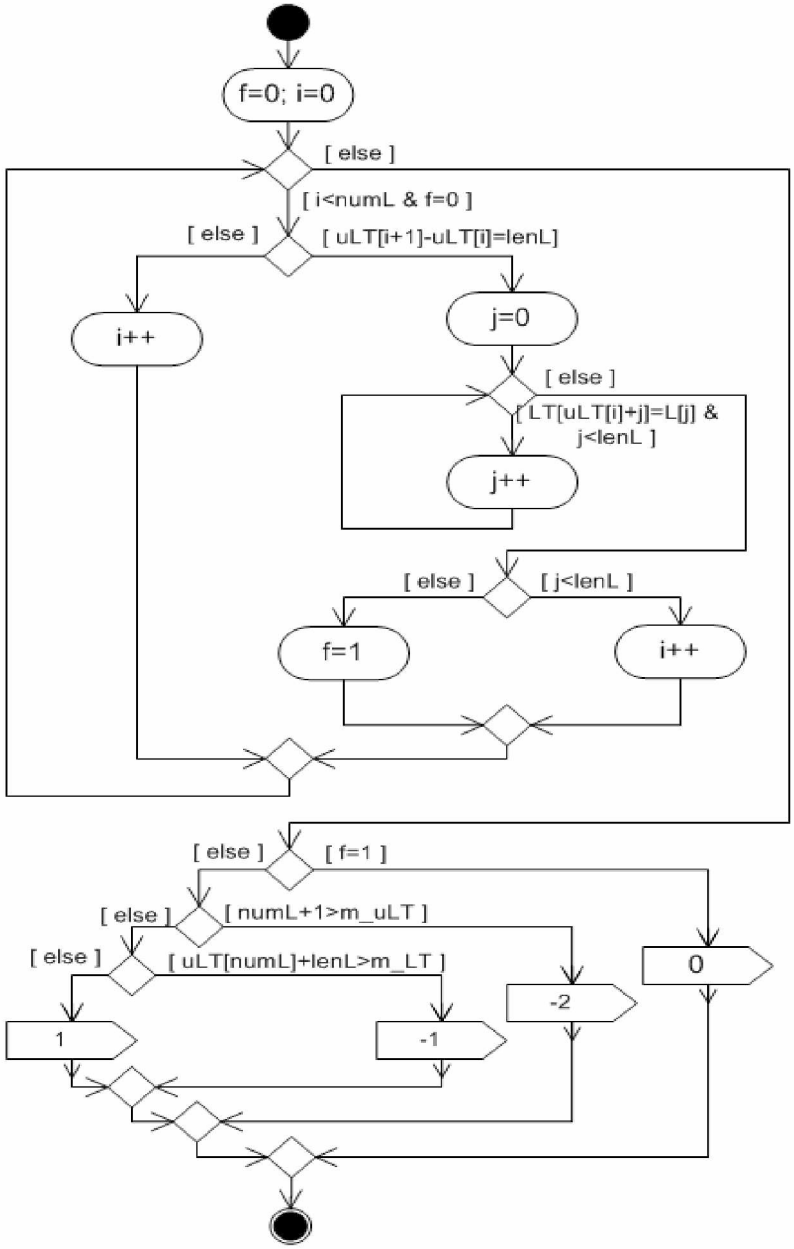


Рис.2.7. Алгоритм работы функции SearchL()

Функция AddL() добавляет линию L в структуру LT, uLT. Также в автомате распознаются следующие ситуации:

- 0 - не найден начальный код линии аннотации;
- 1 - ошибок нет, линии аннотации выделены, структура LT, uLT сформирована;
- -3 - для массива L требуется больший размер.

Алгоритм работы функции AddL() представлен диаграммой деятельности на рис. 2.8.

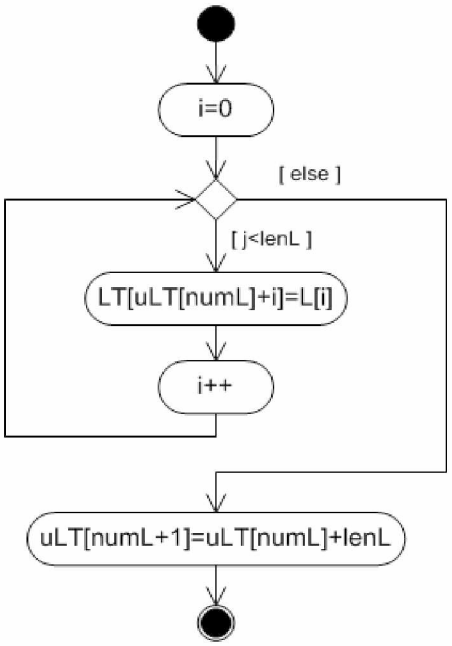


Рис.2.8. Алгоритм работы функции AddL()

После того как все линии ЭКГ найдены, пользователю предоставляется возможность выбрать интересующую его линию. После осуществления данного выбора формируется список начальных отсчетов выбранной линии. Эти действия осуществляет автомат представленный на рис.2.9. Номер выбранной линии записан в переменной nL. Список начальных отсчетов хранится в массиве LS размера lenLS. В автомате распознаются следующие ситуации:

- 0 - линия аннотации с номером nL не существует;

- 1 - ошибок нет, массив LS сформирован;
- -4 - для массива LS требуется больший размер.

На основе данного автомата создается соответствующая функция.

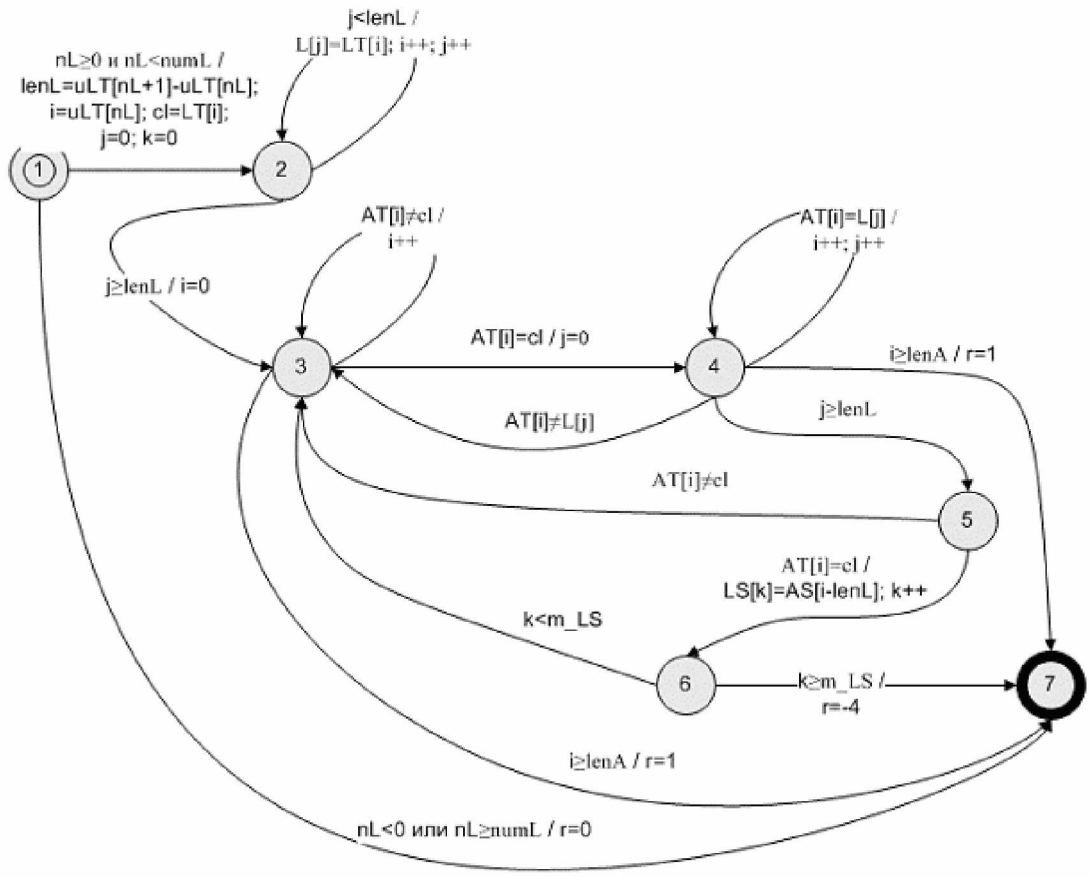


Рис.2.9. Диаграмма состояний конечного автомата формирующего список начальных отсчетов выбранной линии

После того, как список начальных отсчетов выбранной линии сформирован, пользователю предоставляется возможность перемещаться по данным отсчетам с целью поиска характерных участков ЭКГ и их анализа.

ГЛАВА 3. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ

3.1. Разработка функциональной схемы и панели инструментов

Основные интерфейсные элементы системы расположены на панели главного виртуального прибора LabVIEW (на главном экране), которая представлена на рис.3.1. Наиболее значимые элементы помечены соответствующими ярлычками.

Сценарий работы с приложением состоит в выборе файла ЭКГ и просмотре участков ЭКГ с целью формирования врачебного заключения (диагноза).

Для выбора файла ЭКГ используется элемент (1) на рис.3.1, т.е. элемент, помеченный ярлычком с числом 1 на рис.3.1. Далее используются аналогичные обозначения.

Кроме выбора файла также выбирается номер канала ЭКГ. После нажатия на кнопку (3) данные из выбранного файла загружаются в память компьютера и отображаются в окне (5) в виде графика. Также отображаются имя загруженного файла (4), число каналов, номер выбранного канала, время снятия ЭКГ, число отсчетов, разрядность, частота и амплитуда сигнала ЭКГ.

В любой момент работы можно отобразить на графике произвольный участок (окно) ЭКГ. Для этого с помощью курсоров (6) необходимо выбрать границы окна и нажать кнопку (10). При этом старое окно запоминается. К нему можно вернуться, нажав на кнопку «Предыдущее окно». Вернуться назад можно с помощью кнопки «Следующее окно». Кнопка «Начальное окно» служит для удаления всех запомненных окон и возврата к начальному окну.

После считывания данных ЭКГ выполняется автоматическое аннотирование. Также выделяются точки RRS, представляющие собой аннотированные точки для перемещения по комплексам QRS.

При выполнении аннотирования используется набор фильтров. Фильтры хранятся в отдельном каталоге в виде файлов. Набор фильтров может быть изменен при конфигурировании программы. Заданный каталог фильтров отображается на главном экране (2).

Рис.3.1. Главный экран приложения

Номер отсчета сигнала ЭКГ каждой аннотированной точки заносится в массив, который отображается на главном экране (11). Число элементов данного массива соответствует числу аннотированных точек в выбранном окне ЭКГ. Это число также отображается на главном экране. Перемещение по массиву соответствует перемещению по аннотированным точкам ЭКГ. На графике ЭКГ, на текущую аннотированную точку устанавливается курсор (9).

Для этой точки также выводится ее номер, время от начала съема ЭКГ, код аннотации и сама аннотация.

Перемещение по точкам RRS осуществляется аналогично. Отличие состоит в том, что массив точек RRS выводится на главном экране с помощью элемента (12), и на графике ЭКГ, на текущую точку RRS устанавливается курсор (8).

Отметим, что ряд перечисленных функций были разработаны в выпускных квалификационных работах предыдущего года. В данной работе были добавлены функции поиска точек RRS и специфических участков ЭКГ. Другими словами была выполнена модификация подсистемы автоматического аннотирования ЭКГ и выполнена разработка подсистемы просмотра ЭКГ. Более детально выполненная модификация подсистемы автоматического аннотирования ЭКГ будет рассмотрена в разд 3.3, а детали разработки подсистемы просмотра ЭКГ будут рассмотрены в разд.3.4.

Для поиска специфических участков (линий) ЭКГ требуется выбрать начальный код линии (13). После такого выбора осуществляется автоматический поиск линий. Найденные линии отображаются в массиве (14), также отображается число найденных линий.

Линия может встречаться в сигнале ЭКГ несколько раз. Если выбрать интересующую линию, то номера отсчетов сигнала ЭКГ начальных точек выбранной линии отобразятся в массиве (15). Также отобразятся номер текущей начальной точки и число линий. Перемещение по начальным точкам выбранной линии осуществляется аналогично перемещению по точкам аннотации и точкам RRS. На начальную точку выбранной линии устанавливается курсор (7).

На рис.3.1 отображено окно ЭКГ на котором видны две линии с начальным кодом «t)». Выбрана линия «t) (p P p) N Q R s) (t T». На графике ЭКГ курсор (7) установлен на начальную точку этой линии с номером 33230. Время данной точки от начала съема ЭКГ равно 259,609 с. Вторая линия «t) (p P

p) N Q R) (t T», отличается от первой тем, что зубец s отсутствует (Б). В первой линии зубец s есть (А). Заметить такие участки ЭКГ без наличия средств поиска линий было бы затруднительно.

Итак, мы рассмотрели панель инструментов главного виртуального прибора. В LabVIEW программе используется еще 18 разработанных или модифицированных виртуальных прибора. Иерархия приборов представлена на рис.3.2. В данной работе был модифицирован прибор Calc ECG, и были разработаны приборы Aline и S Line.

Прибор Calc ECG реализует функции автоматического аннотирования ЭКГ, приборы Aline и S Line используются при реализации функции просмотра ЭКГ. Прибор Aline осуществляет формирования линий ЭКГ. Прибор S Line осуществляет поиск линий.

Вызовы приборов Calc ECG, Aline отмечены цифрами 1 и 2 на фрагменте функциональной диаграммы главного прибора, представленной на рис.3.3.

Вызов S Line показан на функциональной диаграмме главного прибора представленного на рис. 3.4.

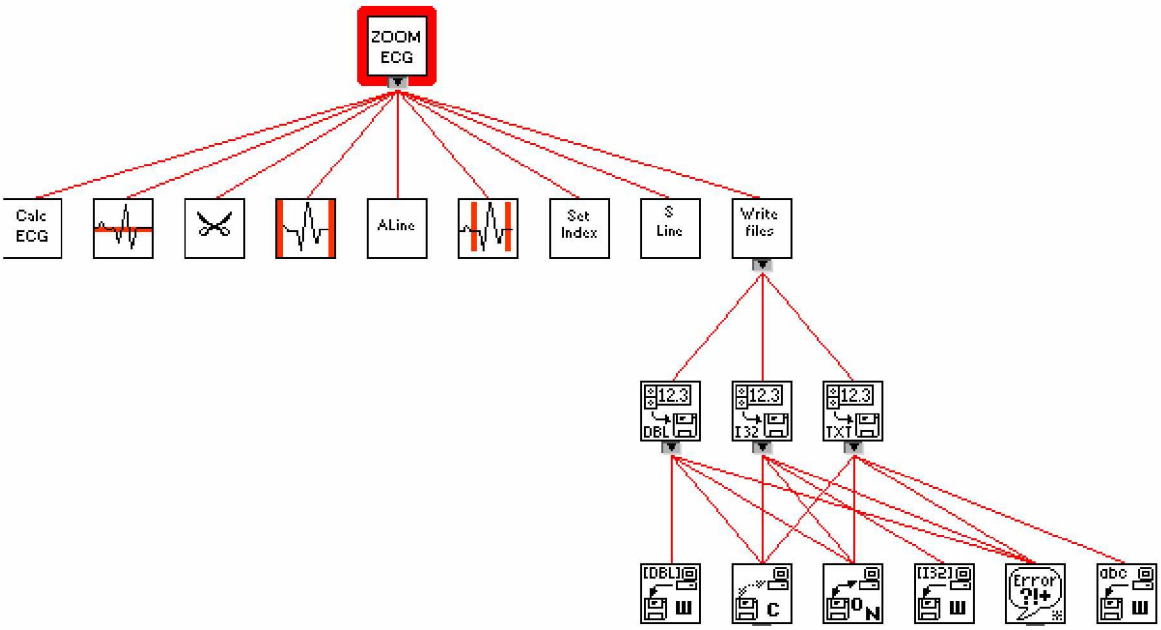


Рис. 3.2. Иерархия виртуальных приборов

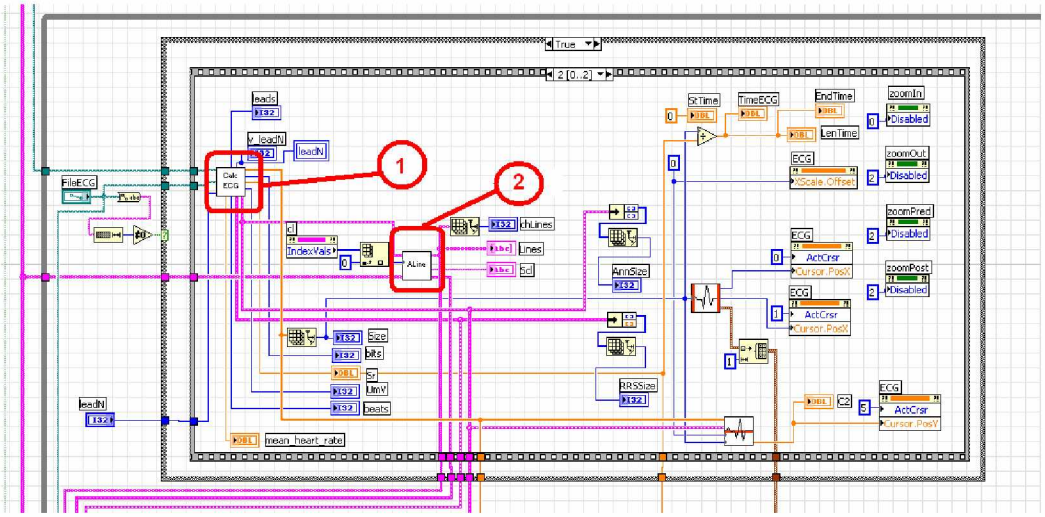


Рис.3.3. Фрагмент функциональной диаграммы главного прибора с блоками аннотирования и формирования линий ЭКГ

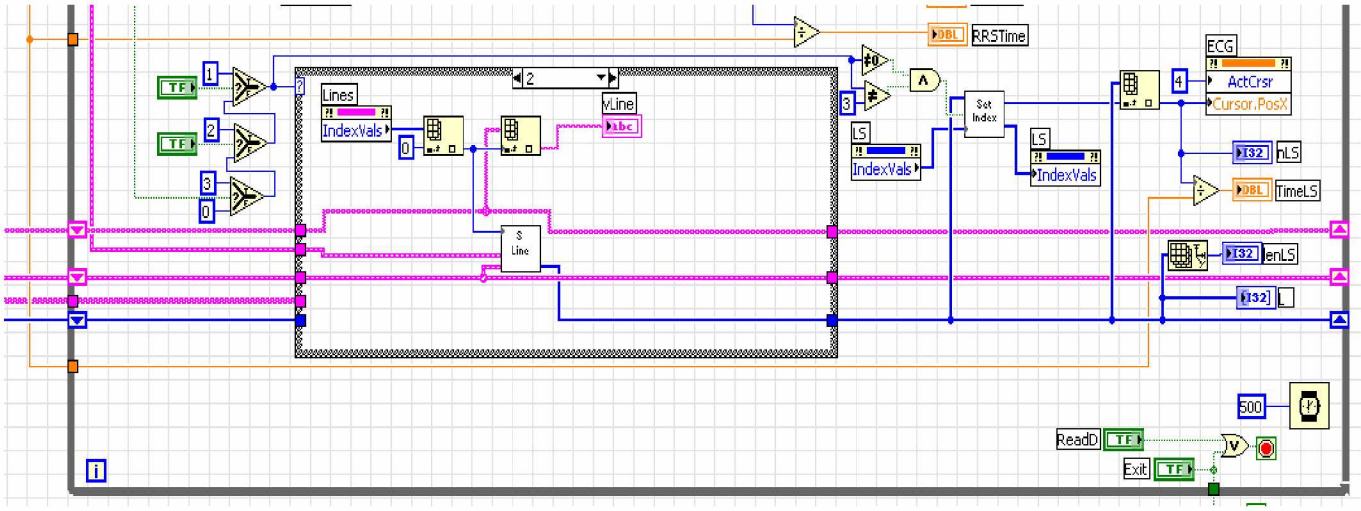


Рис.3.4. Фрагмент функциональной диаграммы главного прибора с блоком поиска линий

В следующих разделах рассмотрим более детально вопросы реализации подсистемы автоматического аннотирования и просмотра ЭКГ.

3.2. Модификация подсистемы автоматического аннотирования ЭКГ

Выбранная в разделе 2.2 библиотека ECG Annotation C++ Library, содержит функции способные распознать 44 особых точки ЭКГ. С особыми точками ЭКГ связываются некоторые цепочки символов, раскрывающие смысловое содержание аннотации в особой точке ЭКГ. Каждая цепочка имеет уникальный код, поэтому можно считать, что с каждой особой точкой ЭКГ связан некоторый символ аннотации с данным кодом. Аннотации и соответствующие цепочки символов сведены в таблицу 3.1.

Таблица 3.1

Коды аннотации

№	Цепочка	Аннотация	№	Код	Аннотация
1.	NotQRS	Нет QRS комплекса	23.	STCH	ST изменение
2.	N	Нормальный ритм	24.	PACESP	Блокада
3.	LBBB	Блокада левой ножки пучка Гиса	25.	T	Зубец Т
4.	RBBB	Блокада правой ножки пучка Гиса	26.	RTM	Изменения ритма
5.	ABERR	Ранняя предсердная экстрасистола	27.	LEARN	Обучение
6.	PVC	Желудочковая экстрасистола	28.	FLWAV	Волны трепетание желудочков
7.	FUSION	Слияние желудочкового и нормального ритма	29.	VFON	Начало фибрилляции
8.	NPC	Узловая экстрасистола	30.	VFOFF	Конец фибрилляции
9.	APC	Предсердная экстрасистола	31.	AESC	Предсердный выскальзывающий импульс
10.	SVPB	Преждевременная или эктопическая суправентрикулярная экстрасистола	32.	SVESC	Суправентрикулярный выскальзывающий импульс
11.	VESC	Желудочковый ритм	33.	NAPC	Нет проведения зубца Р
12.	NESC	Узловой ритм	34.	PFUSE	Переход в сунусовый ритм
13.	PACE	Ритм	35.	FLWAV	Волны трепетание желудочков
14.	Q	Зубец Q	36.	RONT	Экстрасистолия по типу R на T
15.	ARFCT	Изолированный QRS комплекс	37.	(p	Начало зубца Р
16.	STCH	ST изменение	38.	p)	Конец зубца Р
17.	TCH	Изменение волны Т	39.	(t	Начало зубца Т

Таблица 3.1. Продолжение

№	Цепочка	Аннотация	№	Код	Аннотация
18.	SYSTOLE	Систола	40.	t)	Конец зубца Т
19.	DIASTOLE	Диастола	41.	ECT	Электрокардиостимулятор
20.	MEASURE	Параметр аннотации	42.	R	Зубец R
21.	P	Зубец Р	43	S	Зубец S
22.	BBB	Левая или правая ножки пучка Гиса	44.	RONT	Экстрасистолия по типу R на Т

Библиотека ECG Annotation C++ Library представляет собой набор классов. В работе создается функциональная библиотека, которая получается путем создания «объвязки» для библиотеки ECG Annotation C++ Library. Т.е. на основе использования элементов библиотеки ECG Annotation C++ Library создаются функции, которые выполняют считывание данных ЭКГ в формате Physionet и осуществляют автоматическое аннотирование ЭКГ. Создать функциональную библиотеку необходимо для вызова функций из LabVIEW. Данная библиотека оформляется в виде dll.

Функции создаваемой библиотеки оперируют с данными, рассмотренными в разделе 2.4.

Как было отмечено выше, за выполнение автоматического аннотирования ЭКГ отвечает прибор Calc ECG. В данном приборе также реализованы функции считывания ЭКГ из файлов в формате Physionet. Функциональная схема прибора представлена на рис.3.5.

Входными данными являются:

DirFilters – имя каталога с фильтрами, которые используются при аннотировании ЭКГ.

FileECG – имя файла ЭКГ.

LeadN – номер канала ЭКГ.

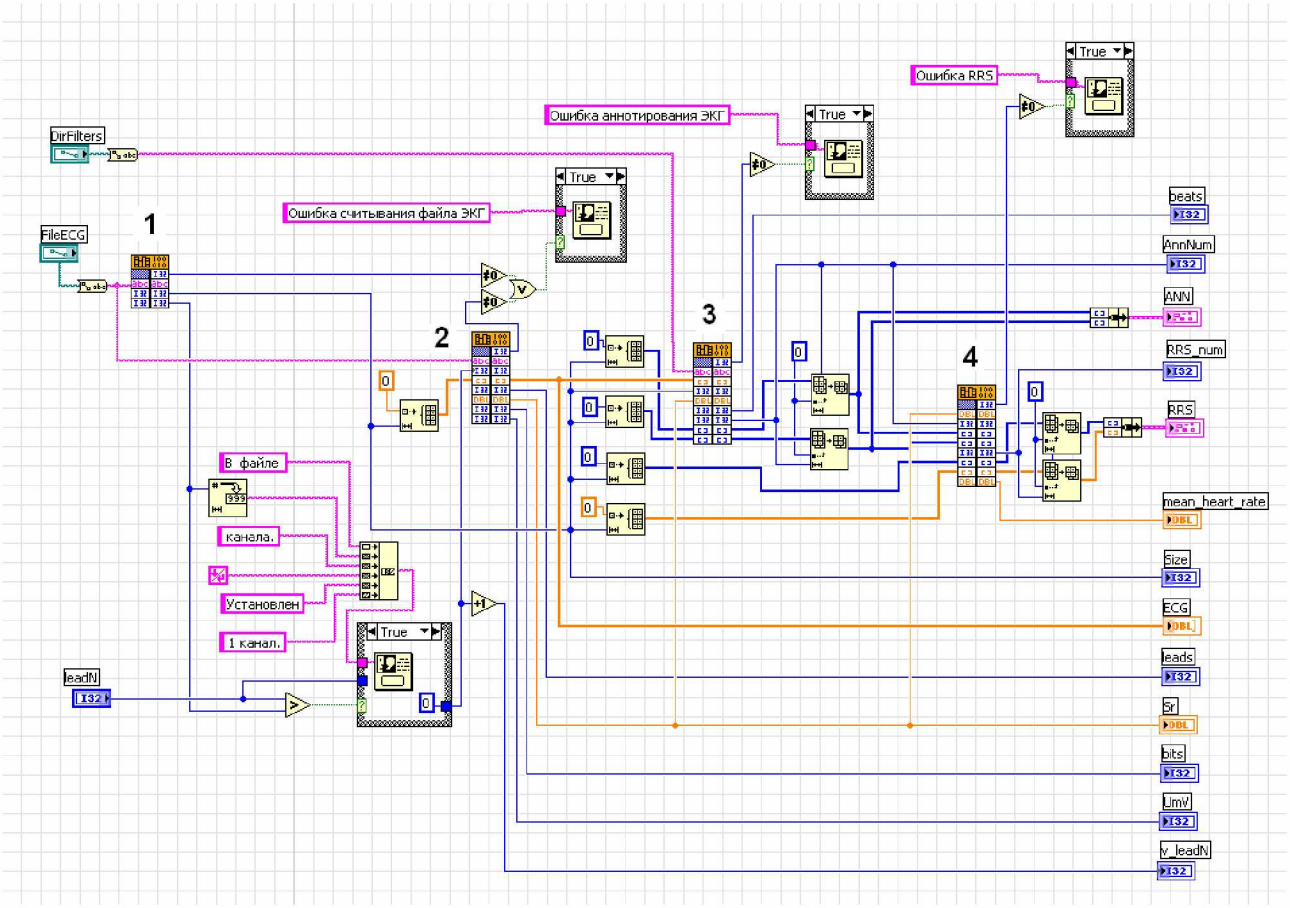


Рис.3.5. Функциональная диаграмма блока считывания и аннотирования ЭКГ

Вначале работы с помощью Call Library Function Node, отмеченного на рис.3.5 цифрой 1, вызывается функция ReadSignalSize. Данная функция определяет размер (число отсчетов) и число каналов ЭКГ, записанной в заданном файле ЭКГ. Сигнатура функции следующая: int ReadSignalSize(const char* nameFileEcg, int* SignalSize, int* leads), где

Вход: const char* nameFileEcg - имя файла ЭКГ.

Выход:

- int* SignalSize - размер сигнала (число отсчетов сигнала в каждом канале ЭКГ).
- int* leads - число каналов ЭКГ.

Функция возвращает:

- 0 - Ошибок нет.
- -1 - Ошибка считывания файла ЭКГ.

Вызываемые функции: ConvNameFile.

Функция использует следующие элементы библиотеки ECG Annotation C++ Library:

- signal.ReadFile,
- signal.GetLength,
- signal.GetLeadsNum.

Далее в LabVIEW инициализируется массив D (см. схему данных на рис.2.5) из SignalSize элементов типа double и с помощью Call Library Function Node, отмеченного на рис.3.5 цифрой 2, вызывается функция ReadSignal. Данная функция считывает сигнал ЭКГ, по заданному каналу из заданного файла ЭКГ. Сигнатура функции следующая: int ReadSignal(const char* nameFileEcg, int leadNumber, double* D, int* leads, double* sr, int* bits, int* UmV) , где

Вход:

- const char* nameFileEcg - имя файла ЭКГ.
- int leadNumber- номер канала.

Выход:

- double* D - сигнал ЭКГ.
- int* leads - число каналов ЭКГ.
- double* sr - частота оцифровки ЭКГ (Гц).
- int* bits - разрядность АЦП.
- int* UmV - амплитуда сигнала ЭКГ(mV).

Функция возвращает:

- 0 - Ошибок нет.
- -2 - Ошибка считывания файла ЭКГ.

Вызываемые функции: ConvNameFile.

Функция использует следующие элементы библиотеки ECG Annotation C++ Library:

- signal.ReadFile,
- signal.GetLength,
- signal.GetLeadsNum,
- signal.GetSR,
- signal.GetBits,
- signal.GetUmV,
- signal.GetData.

После того как сигнал ЭКГ считан (заполнен массив D), выделяется память под массивы AT и AS (см, схему данных на рис. 2.4) и с помощью Call Library Function Node, отмеченного на рис.3.5 цифрой 3, вызывается функция AnnECG. Данная функция выполняет аннотирование ЭКГ, заданной в массиве D. Сигнатура функции следующая: int AnnECG(const char* nameDirFilters, double* D, int SignalSize, double sr, int* beats, int* ANN_num, int* AS, int* AT), где

Вход:

- const char* nameDirFilters- имя каталога фильтров.
- double* D - сигнал ЭКГ.
- int SignalSize - размер сигнала (число отсчетов сигнала в каждом канале ЭКГ).
- double sr - частота оцифровки ЭКГ (Гц).

Выход:

- int* beats- Сердечный ритм.
- int* ANN_num - Число аннотированных точек ЭКГ (размер массива AS и AT).
- int* AS - Номера аннотированных точек ЭКГ в массиве D (номера аннотированных отсчетов ЭКГ).
- int* AT - Коды аннотации в аннотированных точках ЭКГ.

Функция возвращает:

- 0 - Ошибок нет.
- -3 - Ошибка определения QRS-комплекса.

Вызываемые функции: ConvNameFile.

Функция использует следующие элементы библиотеки ECG Annotation C++ Library:

- ann.GetQRS,
- ann.GetQrsNumber,
- ann.GetEctopics,
- ann.GetPTU,
- ann.GetEcgAnnotationSize,
- ann.GetQrsNumber.

После того как сигнал сформирована аннотация ЭКГ (заполнены массивы AT, AS), выделяется память под массивы RRS и RRS_smpl, назначение которых аналогично назначению массивов AT и AS соответственно. Далее с помощью Call Library Function Node, отмеченного на рис.3.5 цифрой 4, вызывается функция RRS_ECG. Данная функция полностью разработана в рамках данной работы. Ее назначение – выделение точек RRS. Сигнатура функции следующая: int RRS_ECG(double sr, int ANN_num, int* AS, int* AT, int* RRS_num, int* RRS_smpl, double* RRS, double* mean_heart_rate),где

Вход:

- double sr - частота оцифровки ЭКГ (Гц).
- int ANN_num - Число аннотированных точек ЭКГ (размер массива ANN_smpl и ANN_type).
- int* AS - Номера аннотированных точек ЭКГ в массиве data (номера аннотированных отсчетов ЭКГ).
- int* AT - Коды аннотации в аннотированных точках ЭКГ.

Выход:

- int* RRS_num - Число точек RRS (размер массива RSS_smpl и RRS).

- int* RRS_smpl - Номера точек RRS в массиве data (номера отсчетов ЭКГ).
- double* RRS - Значение в точке RRS.
- double* mean_heart_rate

Функция возвращает:

- 0 - Ошибок нет,
- -4 - Ошибка определения RRS.

Используемые библиотечные элементы:

- ann.GetRRseq,
- signal.Mean.

3.3. Разработка подсистемы просмотра ЭКГ

В предыдущем разделе была рассмотрена подсистема автоматического аннотирования ЭКГ. Отметим, что аннотация ЭКГ сохраняется в виде двух массивов AT и AS размера lenA. Что число аннотированных точек до выполнения автоматического аннотирования неизвестно. В то же время число таких точек меньше, чем число отсчетов электрокардиосигнала, т.е. lenA <lenD. Поскольку максимальное число элементов массивов AT и AS известно, то память под массивы AT и AS резервировалась в LabVIEW. Размер памяти равнялся lenD. После выполнения аннотирования AnnECG, становился известным реальный размер массивов, т.е. число аннотированных точек ЭКГ (ANN_num). Далее массивы AT и AS усекались до ANN_num ячеек. Аналогичные действия выполнялись для массивов RRS, RRS_smpl.

Такой подход к управлению памятью вполне оправдан, т.к. размеры массивов AT, AS, RRS, RRS_smpl достаточно большие, а их формирование осуществляется только после выбора файла ЭКГ.

В данном разделе рассматривается подсистема просмотра ЭКГ. Система содержит функции расширенного поиска линий ЭКГ. При проектировании этих функций в разд.2.4 были разработаны структуры данных и автоматные модели.

Структуры данных представляют собой массивы LT и uLT для хранения линий ЭКГ и массив LS для хранения начальных отсчетов выбранной линии. Размер массива uLT равен numL+1, где numL – число линий. Размер массива LT равен uLT [numL+1]-1. Размер массива LS равен lenLS.

Отметим, что максимальные размеры массивов LT, uLT, LS число равны числу аннотированных точек lenA (размер массива AT и AS), но реальные размеры, как правило, значительно меньше.

Поэтому резервировать память максимального размера, а затем усекать ее до реального размера средствами LabVIEW не рационально. Можно использовать средства динамического выделения памяти в самих функциях, осуществляющих формирование массивов LT, uLT, LS. Однако, это тоже не является удачным решением, т.к. LabVIEW использует свой менеджер памяти Labview Memory Manager. Поэтому для выделения памяти внутри функций потребуется использовать средства, предоставляемые менеджером памяти LabVIEW. Это приведет к тому, что потеряется универсальность разрабатываемой функциональной библиотеки, т.е. библиотека будет ориентирована только на вызов из среды LabVIEW.

Исходя из этого, было принято решение создать функции, которые для всех возможных ситуаций нехватки памяти возвращают уникальные коды. Функция вызывается из среды LabVIEW с помощью Call Library Function Node и после завершения работы функции анализируется возвращаемый код. Если возникла ошибка, связанная с нехваткой памяти, то в LabVIEW выделяется больше памяти и функция вызывается повторно. Для демонстрационной функции pr_mem10() такая схема проиллюстрирована на рис.3.6. Текст функции следующий:

Листинг 3.1. Функция возвратов уникальных кодов.

```
int pr_mem10(int* arr,int *len) {
if (*len<10) return -1;
*len=10;
for(int i=0;i<*len;i++) arr[i]=i;
return 0; }
```

При первом вызове функции pr_mem10() возникает ошибка, т.к. входной массив состоит из 7 элементов, а требуется 10. Перед вторым вызовом функции размер массива увеличивается на 7 и ошибка не возникает. На выход выдается массив A1=(0,1,2,3,4,5,6,7,8,9,0,0,0,0). Далее в вызывающей программе на LabVIEW из этого массива удаляются лишние элементы, и получается массив A2=(0,1,2,3,4,5,6,7,8,9).

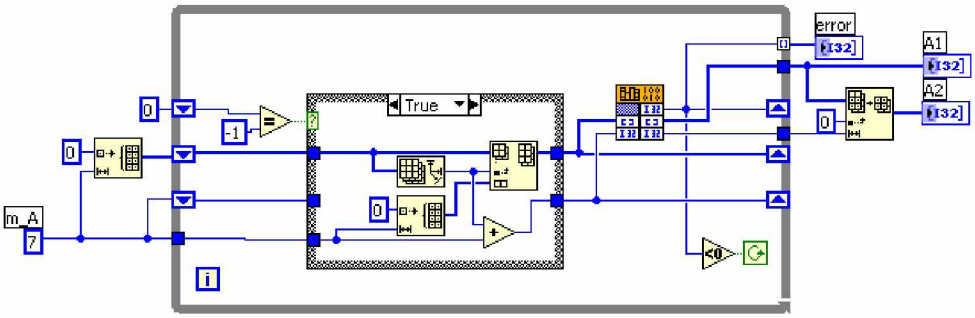


Рис.3.6. Схема вызова функций с учетом обработки аварийных ситуаций связанных с нехваткой памяти

Отметим, что ситуации, связанные с нехваткой памяти, крайне редки. Использование же статических структур позволяет повысить скорость обработки данных и отказаться от механизмов Labview Memory Manager, что в свою очередь позволяет создать независимые от LabVIEW библиотеку функций.

Функциональная диаграмма блока выделения линий ЭКГ представлена на рис.3.7.

На диаграмме видно, как используется рассмотренный прием для выделения памяти под массивы LT, uLT. Первоначальный размер массивов LT и uLT определен константами m_LT и m_uLT соответственно. Аналогично выделяется память под рабочий массив L. Его первоначальный размер определен константой m_L.

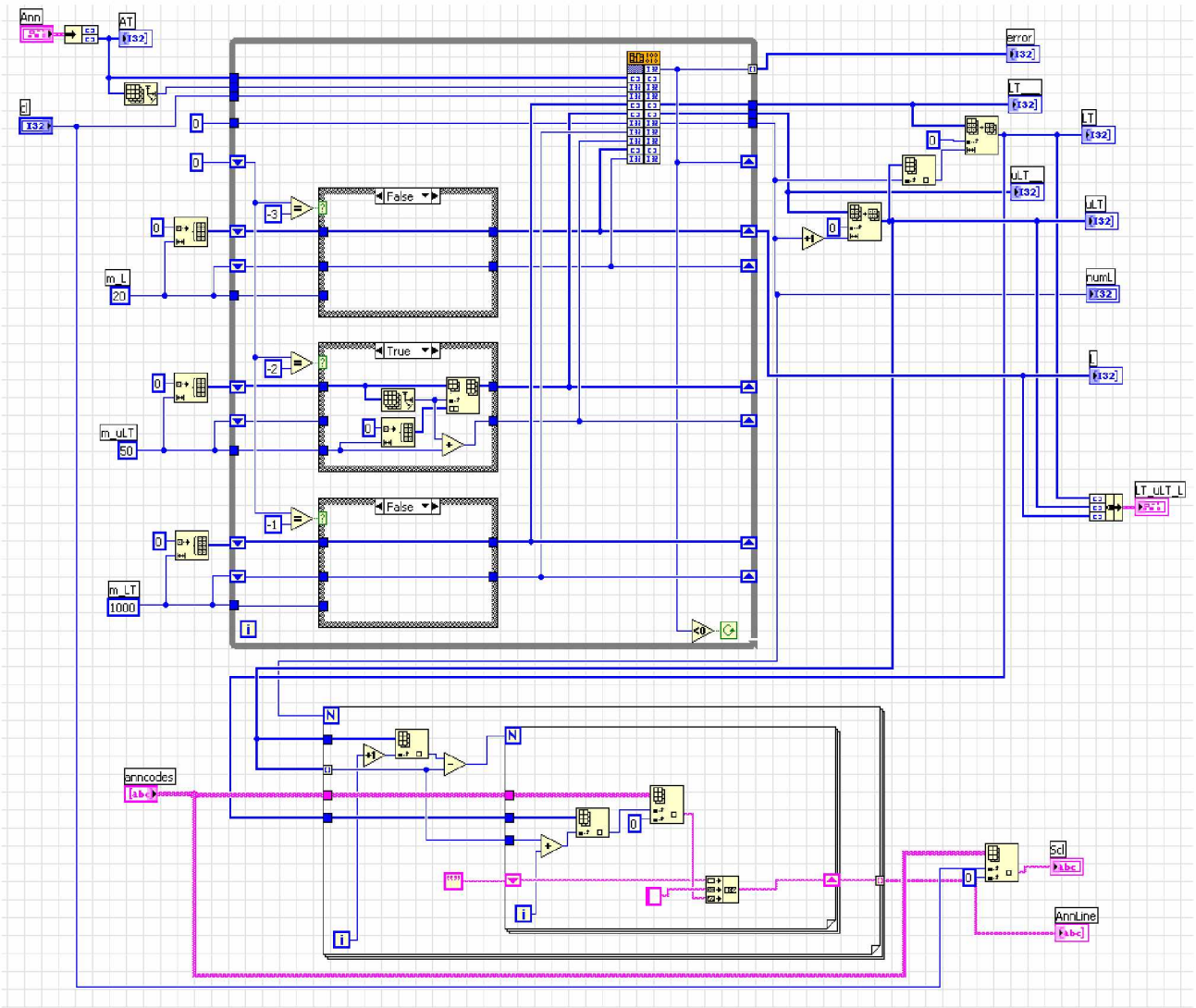


Рис.3.7. Функциональная диаграмма блока выделения линий ЭКГ

После того как память выделена, вызывается функция Aline, осуществляющая формирование линий аннотации ЭКГ с заданным начальным кодом аннотации. Данная функция реализована в в автоматном стиле.

Диаграмма автомата представлена в разд.2.4 на рис.2.5. Сигнатура функции следующая: int ALine(int* AT, int lenA, int cl, int* LT, int* uLT, int* numL, int m_LT, int m_uLT, int* L, int m_L), где:

Вход:

- int* AT - Коды аннотации в аннотированных точках окна ЭКГ.
- int lenA - Число аннотированных точек окна ЭКГ (размер массива AT).
- int cl - Начальный код линии аннотации (если cl==-1, то по умолчанию cl=44 - код "(t").
- int m_LT - Максимальный размер массива LT.
- int m_uLT - Максимальный размер массива uLT.
- int* L - Текущая линия аннотации.
- int m_L - Максимальный размер массива L.

Выход:

- int* LT - Уникальные линии аннотации (последовательность кодов аннотации) хранящиеся в гнездовой структуре.
- int* uLT - Указатели (курсоры) на начало гнезд в массиве LT.
- int* numL - Число уникальных линий аннотации равно (размер массива uLT равен numL+1).

Функция возвращает:

- 0 - Не найден начальный код линии аннотации.
- 1 - Ошибок нет, линии аннотации выделены.
- -1 - Для массива LT требуется размер больше, чем m_LT.
- -2 - Для массива uLT требуется размер больше, чем m_uLT.
- -3 - Для массива L требуется размер больше, чем m_L.

Вызываемые функции:

- SeachL,
- AddL.

Функция Aline вызывает функции SeachL и AddL. Алгоритмы работы этих функций были разработаны в разд 2.4.

Функция SeachL определяет уникальность заданной линии аннотации ЭКГ и проверяет возможность ее добавления в структуру LT, uLT. Сигнатура функции следующая: int SeachL(int LT[], int uLT[], int *numL, int L[], int lenL, int m_uLT, int m_LT), где:

Вход:

- int* LT - Уникальные линии аннотации (последовательность кодов аннотации) хранящиеся в гнездовой структуре.
- int* uLT - Указатели (курсоры) на начало гнезд в массиве LT.
- int* numL - Число уникальных линий аннотации равно (размер массива uLT равен numL+1).
- int* L - Текущая линия аннотации.
- int lenL - Длина текущей линии аннотации.
- int m_uLT - Максимальный размер массива uLT.
- int m_LT - Максимальный размер массива LT.

Функция возвращает:

- 0 - линия аннотации не уникальна.
- 1 - линия аннотации уникальна и может быть добавлена функцией AddL в гнездовую структуру (массивы LT, uLT).
- -1 - линия аннотации уникальна, но для ее добавления в гнездовую структуру нет места в массиве LT.
- -2 - линия аннотации уникальна, но для ее добавления в гнездовую структуру нет места в массиве uLT.

Функция AddL добавляет линию аннотации ЭКГ в структуру LT, uLT. Сигнатура функции следующая: void AddL(int LT[], int uLT[], int *numL, int L[], int lenL), где:

Вход:

- int* L - Текущая линия аннотации.
- int lenL - Длина текущей линии аннотации.

Выход:

- int* LT - Уникальные линии аннотации (последовательность кодов аннотации) хранящиеся в гнездовой структуре.
- int* uLT - Указатели (курсоры) на начало гнезд в массиве LT.
- int* numL - Число уникальных линий аннотации равно (размер массива uLT равен numL+1).

Функциональная диаграмма блока поиска линий представлена на рис. 3.8. Для перемещения по выделенным линиям ЭКГ формируется массив LS для хранения начальных отсчетов выбранной линии. На диаграмме видно, что для выделения памяти под массив LS используется рассмотренный прием. Первоначальный размер массива определен константой m_LS.

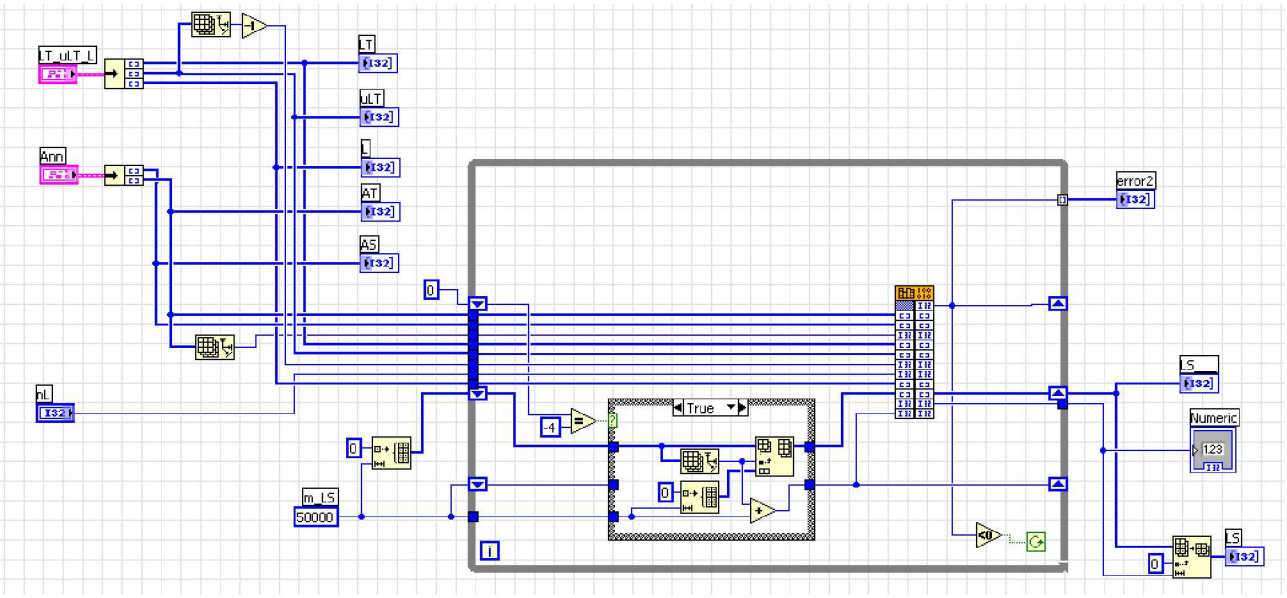


Рис.3.8. Функциональная диаграмма блока поиска линий

Для формирования массива LS, т.е. для нахождения номеров отсчетов первых аннотированных точек линии с заданным номером используется функция Sline. Функция SLine реализована в автоматном стиле. Соответствующий автомат был рассмотрен в разд.2.4. Сигнатура функции следующая: int SLine(int* AT, int* AS, int lenA, int* LT, int* uLT, int numL, int nL, int* L, int* LS, int* lenLS, int m_LS), где:

Вход:

- int* AT - Коды аннотации в аннотированных точках окна ЭКГ.
- int* AS - Номера аннотированных точках окна ЭКГ.
- int lenA - Число аннотированных точек окна ЭКГ (размер массива AT и AS).
- int* LT - Уникальные линии аннотации (последовательность кодов аннотации) хранящиеся в гнездовой структуре.
- int* uLT - Указатели (курсоры) на начало гнезд в массиве LT.
- int numL - Число уникальных линий аннотации равно (размер массива uLT равен numL+1).
- int nL - Номер линии аннотации (0<=nl<numL numL+1 - размер массива uLT).
- int* L - Текущая линия аннотации.
- int m_LS - Максимальный размер массива LS.

Выход:

- int* LS - Номера первых аннотированных точек линии с номером nL.
- int* lenLS - Размер массива LS.

Функция возвращает:

- 0 - Линия аннотации с номером nL не существует (массив LS - пустой).
- 1 - Ошибок нет, номера первых точек линии аннотации с номером nL выделены.
- -4 - Для массива LS требуется размер больше, чем m_LS.

3.4. Тестирование программного обеспечения на данных Physionet

Исследовались различные файлы из следующих каталогов банка PhysioNet [4]:

- ANSI-AAMI EC13 Test Waveforms (9 файлов ЭКГ);
- European_ST-T_Database (90 файлов ЭКГ);

- MIT-BIH Arrhythmia Database (2 файла ЭКГ);
- The MIT-BIH Malignant Ventricular Arrhythmia Database (2 файла ЭКГ);
- The QT Database (1 файл ЭКГ);
- T-Wave Alternans Challenge Database (100 файлов ЭКГ);

Файлы содержали записи ЭКГ. Данные каталогов European_ST-T_Database и T-Wave Alternans Challenge Database были считаны из банка PhysioNet полностью. Это обусловлено тем, что ЭКГ из этих каталогов имеют наивысший разброс во времени записи. Так время записи ЭКГ из каталога T-Wave Alternans Challenge Database равно 114 с. (объем файлов 1,31МБ), а время записи ЭКГ из каталога European_ST-T_Database равно 7200 с. (объем файлов 465МБ). Для каждого из файлов ЭКГ производилось аннотирование и формирование линий. При этом средствами LabVIEW замерялось время выполнения каждого действия. Для файлов одинакового размера результаты замеров времени усреднялись. Таким образом, наиболее точно среднее время работы вычислено для крайних точек.

В результате проведенных экспериментов получены две зависимости:

1. Зависимость времени формирования аннотации от числа отсчетов ЭКГ.
2. Зависимость времени формирования линий ЭКГ от числа аннотированных точек.

Первая зависимость представлена на рис. 3.9, вторая – на рис.3.10.

Из графиков видно, что даже для файлов ЭКГ, состоящих из 1800000 отсчетов (7200 с. * 250 Гц = 1800000), время расчета аннотации не превышало 3,1 с. Если учесть, что аннотация рассчитывается только один раз при считывании файла ЭГК, то такое время является вполне допустимым и не приводит к субъективной неудовлетворенности пользователей.

В отличие от расчета аннотации, формирование линий осуществляется каждый раз, когда пользователь вводит начальный символ линии. Но как видно из второго графика время формирования линий не превышает 0,016 с., что также не приводит к субъективной неудовлетворенности пользователей.

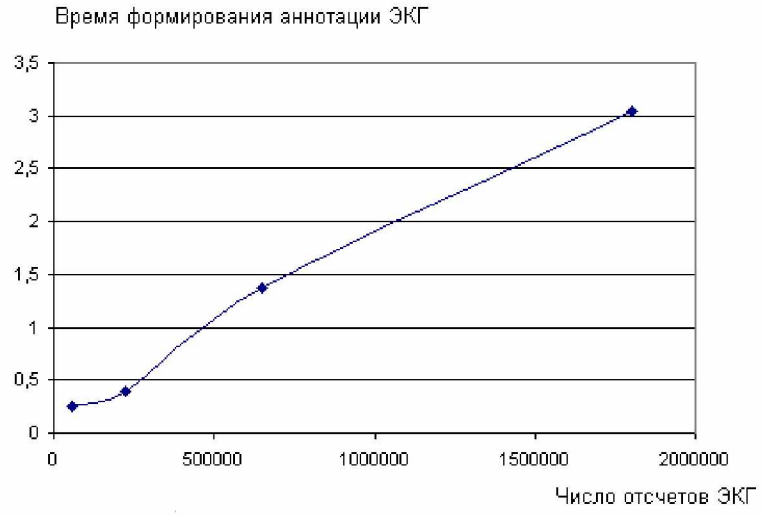


Рис.3.9. Зависимость времени формирования аннотации от числа отсчетов ЭКГ.

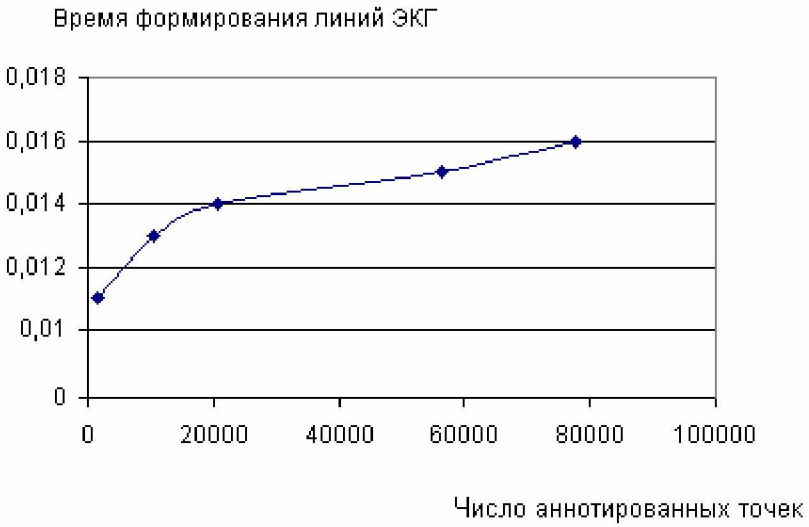


Рис.3.10. Зависимость времени формирования линий ЭКГ от числа аннотированных точек.

Таким образом, можно сделать вывод, что скорость работы программы удовлетворяет требованиям к качественному пользовательскому интерфейсу.

ЗАКЛЮЧЕНИЕ

В данной выпускной квалификационной работе была поставлена цель - создать автоматизированную систему анализа ЭКГ с расширенными возможностями поиска диагностически значимых участков.

Система позволяет повысить эффективность работы врачей при оценки состояния сердца.

Для достижения поставленной цели были решены следующие основные задачи:

1. Изучены некоторые вопросы предметной области и выполнена постановку задачи.
2. Разработана структурная схема программного обеспечения.
3. Обоснован выбор библиотеки обработки кардиосигналов.
4. Обоснован выбор инструментальных средств.
5. Разработаны структуры данных.
6. Разработаны модели программного обеспечения. Использовались автоматные модели.
7. Выполнена программная реализация системы.
8. Выполнено тестирование разработанного программного обеспечения на данных банка ЭКГ Physionet.

Таким образом, поставленная цель работы была достигнута.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Муромцев В.В., Никитин В.М., Ефремова О.А., Липунова Е.А., Камышникова Л.А. Программная реализация системы компьютерного анализа ЭКГ с расширенными возможностями автоматизированного поиска характкрных участков. Научный результат, серия Медицина и фармация, №2(8) 2016 г. с.10-16.

2. Olga A. Efremova, Valeriy M. Nikitin, Viktor V. Muromtsev, Elena A Lipunova, Ludmila A. Kamyshnikova. ECG computer analysis system with the advanced features of automated search and identification of it's diagnostically significant changes. - IJPT| June-2016 | Vol. 8 | Issue No.2 | 14174-14181 Page 14174-14181, http://www.ijptonline.com/wp-content/uploads/2016/07/14174-14181.pdf

3. Сизенцева Г.П. Методическое пособие по электрокардиографии.- М.: Издательство НЦССХ им. А.Н.Бакулева РАМН, 1998. - 68 с.

4. PhysioNet the research resource for complex physiologic signals. [Электронный ресурс].- Режим доступа: http://physionet.org

5. Маклаков С.В. Создание информационных систем с AllFusion Modeling Suite.- М.: ДИ,АЛОГ-МИФИ, 2005.- 432 с.

6. Петров С.П., Епишина Е.В., Воронин В.В. Оценка алгоритмов распознавания образов для задач автоматического анализа электрокардиограмм // Евразийский Союз Ученых (ЕСУ). Ежемесячный научный журнал, № 8 / 2014, часть 8, с.27-29.

7. Библиотека аннотации ЭКГ на C++. [Электронный ресурс].- Режим доступа: https://www.codeproject.com/articles/20995/ecg-annotation-c-library

8. National Instruments [Электронный ресурс]. - Режим доступа: http://www.ni.com

9. Пейч Л.И., Точилин Д.А., Поллак Б.П. LabVIEW для новичков и специалистов.- М.: Горячая линия-Телеком, 2004.- 384 с.

10. Суранов А. Я. LabVIEW 8.20: Справочник по функциям. – М.: ДМК Пресс, 2007. – 536 с.

11. Федосов В. П., Нестеренко А. К. Цифровая обработка сигналов в LabVIEW: учеб. пособие / под ред. В. П. Федосова. – М.: ДМК Пресс, 2007. – 456 с.

12. Страуструп Б. Язык программирования С++ Специальное издание. Бином, 2007. – 1054 с.

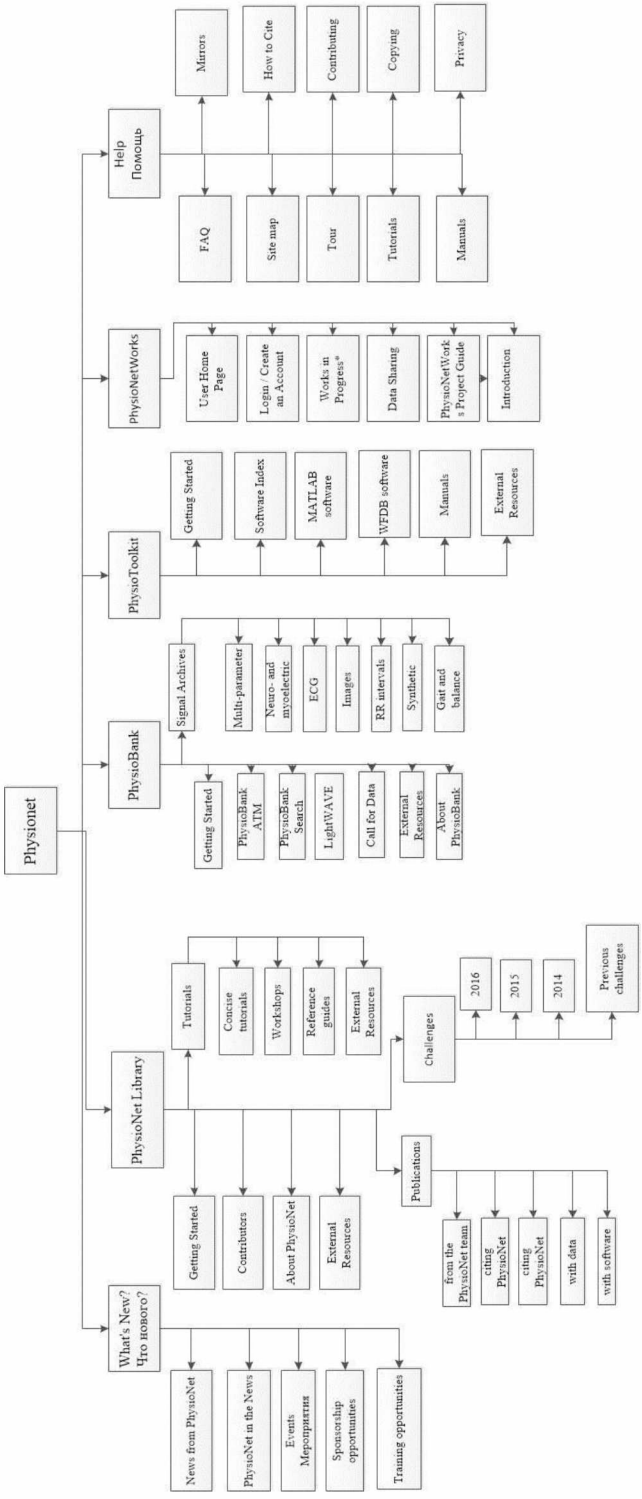
13. Пауэрс Л. Microsoft Visual Studio 2008.- Спбю: БХВ-Петербург,- 2009.- 1200с.

14. Documents associated with Unified Modeling. [Электронный ресурс].- Режим доступа: http://www.omg.org/spec/UML/2.5/

15. Фаулер М. UML.Основы,3-е издание. СПб: Символ-Плюс, 2004-192с.

ПРИЛОЖЕНИЕ 1

Структура ресурса PhysioNet



ПРИЛОЖЕНИЕ 2

```
#ifdef ECG_C_EXPORTS
#define ECG_C_API __declspec(dllexport)
#else
#define ECG_C_API __declspec(dllimport)
#endif

#include "signal.h"
#include "cwt.h"
#include "fwt.h"
#include "ecgannotation.h"
#include "ecgdenoise.h"

extern ECG_C_API
int ReadSignalSize(const char* nameFileEcg,
                           int* SignalSize,
                           int* leads);

extern ECG_C_API
int ReadSignal(const char* nameFileEcg,
                     int leadNumber,
                     double* data,
                     int* leads,
                     double* sr,
                     int* bits,
                     int* UmV
                     );


extern ECG_C_API
int AnnECG(const char* nameDirFilters,
               double* data,
               int SignalSize,
               double sr,
               int* beats,
               int* ANN_num,
               int* ANN_smpl,
               int* ANN_type
               );

extern ECG_C_API
int RRS_ECG(double sr,
               int ANN_num,
               int* ANN_smpl,
               int* ANN_type,
               int* RRS_num,
               int* RRS_smpl,
           double* RRS,
               double* mean_heart_rate
               );



extern ECG_C_API
int ALine(        int* AT,
                        int lenA,
                        int cl,
```

```
                        int* LT,
                        int* uLT,
                        int* numL,
                        int m_LT,
                        int m_uLT,
                        int* L,
                        int m_L
                    )
;

extern ECG_C_API
int ALine_for_compress( int* AT,
                                    int lenA,
                                    int cl,
                                    int* LT,
                                    int* uLT,
                                    int* numL,
                                    int* ATc,
                                    int* lATc,
                                    int* L,
                                    int m_LT,
                                    int m_uLT,
                                    int m_ATc,
                                    int m_L
                    )
;

extern ECG_C_API
int SLine(          int* AT,
                        int* AS,
                        int lenA,
                        int* LT,
                        int* uLT,
                        int numL,
                        int  nL,
                        int* L,
                        int* LS,
                        int* lenLS,
                        int m_LS
                    );
```

ПРИЛОЖЕНИЕ 3

```
// ECG_C.cpp: определяет экспортированные функции для приложения DLL.
//

#include "stdafx.h"
#include "ECG_C.h"


// MY FUNCTION
#include "lib.h"


/*
Функция: ConvNameFile(  const char* name1,
                                    wchar_t* name2)
Назначение: Конвертирует имя файла.
Вход:       const char* name1 - исходное имя файла.
Выход:            wchar_t* name2          - выходное имя файла.
*/
void ConvNameFile(const char* name1,
                          wchar_t* name2)
{
    for (int i = 0; i < (int)strlen(name1); i++)
              mbtowc(name2 + i, name1 + i, MB_CUR_MAX);
}


/*
Функция: int ReadSignalSize(const char* nameFileEcg,
                                          int* SignalSize,
                                          int* leads)
Назначение: Определяет размер размер сигнала и число каналов ЭКГ, записанного в
заданном файле ЭКГ.
Вход:       const char* nameFileEcg - имя файла ЭКГ.
Выход:            int* SignalSize               - размер сигнала (число отсчетов
сигнала в кажном канале ЭКГ).
                  int* leads                    - число каналов ЭКГ.
Возвращает:  0 - Ошибок нет.
                  -1 - Ошибка считывания файла ЭКГ.
Вызываемые функции: ConvNameFile.
Используемые библиотечные элементы: signal.ReadFile, signal.GetLength,
signal.GetLeadsNum.
*/
ECG_C_API
int ReadSignalSize(const char* nameFileEcg,
                           int* SignalSize,
                           int* leads
                           )
{
      wchar_t nameFile[_MAX_PATH] = L"";
      ConvNameFile(nameFileEcg,nameFile);
      class Signal signal;
    if (signal.ReadFile(nameFile)) {
       *(SignalSize)=signal.GetLength();
         *(leads)=signal.GetLeadsNum();
        return 0;  //Ошибок нет
      } else {
```

```
        return -1; //Ошибка считывания файла ЭКГ
      }

}


/*
Функция: ReadSignal(    const char* nameFileEcg,
                                    int leadNumber,
                                    double* data,
                                    int* leads,
                                    double* sr,
                                    int* bits,
                                    int* UmV
                                )
Назначение: Считывает сигнал ЭКГ, по заданному каналу из заданного файла ЭКГ.
Вход: const char* nameFileEcg - имя файла ЭКГ.
            int leadNumber                - номер канала.
Выход:      double* data       - сигнал ЭКГ.
            int* leads         - число каналов ЭКГ.
            double* sr         - частота оцифровки ЭКГ (Гц).
            int* bits          - разрядность АЦП.
            int* UmV           - амплитуда сигнала ЭКГ(mV).
Возвращает:        0 - Ошибок нет.
                       -2 - Ошибка считывания файла ЭКГ.
Вызываемые функции: ConvNameFile.
Используемые библиотечные элементы: signal.ReadFile, signal.GetLength,
signal.GetLeadsNum,
signal.GetSR, signal.GetBits, signal.GetUmV, signal.GetData.
*/
ECG_C_API
int ReadSignal(const char* nameFileEcg,
                    int leadNumber,
                    double* data,
                    int* leads,
                    double* sr,
                    int* bits,
                    int* UmV
                    )
{
      wchar_t nameFile[_MAX_PATH] = L"";
      ConvNameFile(nameFileEcg,nameFile);
      class Signal signal;
    if (signal.ReadFile(nameFile)) {
        int SignalSize=signal.GetLength();
        *(leads)= signal.GetLeadsNum();
      *(sr) = signal.GetSR();
        *(bits)=signal.GetBits();
        *(UmV)=signal.GetUmV();
        double*  pData = signal.GetData(leadNumber);
        for (int i=0; i<SignalSize; i++) data[i]=pData[i];
        return 0;  //Ошибок нет
      } else {
        return -2; //Ошибка считывания файла ЭКГ
      }
}


/*
Функция: int AnnECG(const char* nameDirFilters,
                            double* data,
                            int SignalSize,
                            double sr,
```

```
                              int* beats,
                              int* ANN_num,
                              int* ANN_smpl,
                              int* ANN_type
                            )
Назначение: Аннотирование ЭКГ.
Вход: const char* nameDirFilters    - имя каталога фильтров.
            double* data                          - сигнал ЭКГ.
            int SignalSize                        - размер сигнала (число отсчетов
сигнала в кажном канале ЭКГ).
            double sr                             - частота оцифровки ЭКГ (Гц).
Выход:      int* beats        - Сердечный ритм.
            int* ANN_num      - Число аннотированных точек ЭКГ (размер массива
ANN_smpl и ANN_type).
            int* ANN_smpl     - Номера аннотированных точек ЭКГ в массиве data
(номера аннотированных отсчетов ЭКГ).
            int* ANN_type     - Коды аннотации в аннотированных точках ЭКГ.
Возвращает:        0 - Ошибок нет.
                        -3 - Ошибка определения QRS-комплекса.
Вызываемые функции: ConvNameFile.
Используемые библиотечные элементы: ann.GetQRS, ann.GetQrsNumber,
ann.GetEctopics, ann.GetPTU,
ann.GetEcgAnnotationSize, ann.GetQrsNumber.
*/
ECG_C_API
int AnnECG(const char* nameDirFilters,
               double* data,
               int SignalSize,
               double sr,
               int* beats,
               int* ANN_num,
               int* ANN_smpl,
               int* ANN_type
               )
{
      wchar_t nameDirF[_MAX_PATH] = L"";
      ConvNameFile(nameDirFilters,nameDirF);
  class EcgAnnotation ann;  //default annotation params
                            //or add your custom ECG params to annotation class
                            //from lib.h
                            // ANNHDR hdr;
                            // hdr.minbpm = 30;
                            //  etc...
                            // class EcgAnnotation ann( &hdr );
  class Signal signal;
   int** qrsAnn = ann.GetQRS(data, SignalSize, sr, nameDirF);
    *(beats)=0;
  if (qrsAnn) {
     *(beats)=ann.GetQrsNumber();
       ann.GetEctopics(qrsAnn, ann.GetQrsNumber(), sr);
     int annNum = 0;
     int** ANN = ann.GetPTU(data, SignalSize, sr, nameDirF, qrsAnn,
ann.GetQrsNumber());
     if (ANN) {
        annNum = ann.GetEcgAnnotationSize();
     } else {
        ANN = qrsAnn;
        annNum = 2 * ann.GetQrsNumber();
     }

     //printing out annotation
     for (int i = 0; i < annNum; i++) {
        ANN_smpl[i]=ANN[i][0];
```

```
            ANN_type[i]=ANN[i][1];
        }
        *(ANN_num)=annNum;

  } else {
        return -3;  }
  return 0;
}


/*
Функция: int RRS_ECG(   double sr,
                                    int ANN_num,
                                    int* ANN_smpl,
                                    int* ANN_type,
                                    int* RRS_num,
                                    int* RRS_smpl,
                                    double* RRS,
                                    double* mean_heart_rate
                        )
Назначение: Выделение RRS.
Вход: double sr          - частота оцифровки ЭКГ (Гц).
            int ANN_num       - Число аннотированных точек ЭКГ (размер массива
ANN_smpl и ANN_type).
            int* ANN_smpl     - Номера аннотированных точек ЭКГ в массиве data
(номера аннотированных отсчетов ЭКГ).
            int* ANN_type     - Коды аннотации в аннотированных точках ЭКГ.
Выход:      int* RRS_num      - Число точек RRS (размер массива RSS_smpl и RRS).
            int* RRS_smpl     - Номера точек RRS в массиве data (номера отсчетов
ЭКГ).
            double* RRS       - Значение в точке RRS.
            double* mean_heart_rate
Возвращает:        0 - Ошибок нет,
                        -4 - Ошибка определения RRS.
Используемые библиотечные элементы: ann.GetRRseq, signal.Mean.
*/
ECG_C_API
int RRS_ECG(double sr,
               int ANN_num,
               int* ANN_smpl,
               int* ANN_type,
               int* RRS_num,
               int* RRS_smpl,
           double* RRS,
               double* mean_heart_rate    //
               )
{
  class EcgAnnotation ann;  //default annotation params
                            //or add your custom ECG params to annotation class
                            // from lib.h
                            // ANNHDR hdr;
                            //  hdr.minbpm = 30;
                            //  etc...
                            // class EcgAnnotation ann( &hdr );
  class Signal signal;

   int **ANN;
   ANN = new int* [ANN_num];
   for (int i = 0; i < ANN_num; i++) {
      ANN[i] = new int[3];
      ANN[i][0]=ANN_smpl[i];
      ANN[i][1]=ANN_type[i];
   }
```

```
   //saving RR seq
   vector<double> rrs;
   vector<int> rrsPos;
   if (ann.GetRRseq(ANN, ANN_num, sr, &rrs, &rrsPos)) {
     for (int i = 0; i < (int)rrs.size(); i++) {
       RRS_smpl[i]=rrsPos[i];
       RRS[i]=rrs[i];
     }
     *(RRS_num)=(int)rrs.size();
     *(mean_heart_rate)=signal.Mean(&rrs[0], (int)rrs.size());
  } else {
       return -4; //
  }
  return 0;
}


/*
Функция: int SeachL(int LT[],
                            int uLT[],
                            int *numL,
                            int L[],
                            int lenL,
                            int m_uLT,
                            int m_LT)
Назначение: Определяет уникальность заданной линии аннотации ЭКГ и проверяет
возможность
                 добавления уникальной линии аннотации в структуру гнездового
типа.
Вход: int* LT          - Уникальные линии аннотации (последовательность кодов
аннотации) хранящиеся в гнездовой структуре.
            int* uLT    - Указатели (курсоры) на начало гнезд в массиве LT.
            int* numL   - Число уникальных линий аннотации равно (размер массива
uLT равен numL+1).
            int* L            - Текущая линия аннотации.
            int lenL    - Длина текущей линии аннотации.
a           int m_uLT   - Максимальный размер массива uLT.
            int m_LT    - Максимальный размер массива LT.
Возвращает:  0 - линия аннотации не уникальна.
                  1 - линия аннотации уникальна и может быть добавлена функцией
AddL в гнездовую структуру (массивы LT, uLT).
                 -1 - линия аннотации уникальна, но для ее добавления в
гнездовую структуру нет места в массиве LT.
                 -2 - линия аннотации уникальна, но для ее добавления в
гнездовую структуру нет места в массиве uLT.
*/
int SeachL(int LT[],
               int uLT[],
               int *numL,
               int L[],
               int lenL,
               int m_uLT,
               int m_LT)
{
      int f=0, i=0, j=0;
      while ((i<*numL)&&(f==0))
            if (uLT[i+1]-uLT[i]==lenL) {
                  j=0;
                  while (LT[uLT[i]+j]==L[j]) j++;
                  if (j<lenL) i++; else f=1;
            }
            else i++;
```

```
      if (f==1) return 0;
      if (*numL+2>m_uLT) return -2;
      if (uLT[*numL]+lenL>m_LT) return -1;
      return 1;
}


/*
Функция: void AddL(     int LT[],
                              int uLT[],
                              int *numL,
                              int L[],
                              int lenL
                  )
Назначение: Добавляет линию аннотации ЭКГ в структуру гнездового типа.
Вход: int* L            - Текущая линия аннотации.
            int lenL    - Длина текущей линии аннотации.
Выход:      int* LT           - Уникальные линии аннотации (последовательность
кодов аннотации) хранящиеся в гнездовой структуре.
            int* uLT    - Указатели (курсоры) на начало гнезд в массиве LT.
            int* numL   - Число уникальных линий аннотации равно (размер массива
uLT равен numL+1).
*/

void AddL(int LT[],     //Уникальные линии аннотации (последовательность кодов
аннотации) хранящиеся в гнездовой структуре.
              int uLT[],//Указатели (курсоры) на начало гнезд в массиве LT.
              int *numL,//Число уникальных линий аннотации равно (размер массива
uLT равен numL+1).
              int L[],  //Текущая линия аннотации.
              int lenL  //Длина текущей линии аннотации.
              )
{
      for (int i=0; i<lenL; i++)
            LT[uLT[*numL]+i]=L[i];
      uLT[*numL+1]=uLT[*numL]+lenL;
      *numL=*numL+1;
}

/*
Функция: int ALine(     int* AT,
                              int lenA,
                              int cl,
                              int* LT,
                              int* uLT,
                              int* numL,
                              int m_LT,
                              int m_uLT,
                              int* L,
                              int m_L
                  )
Назначение: Выделение линий аннотации ЭКГ с заданным начальным кодом аннотации.
                  * Функция ALine реализована в автоматном стиле.
Вход: int* AT           - Коды аннотации в аннотированных точках окна ЭКГ.
            int lenA    - Число аннотированных точек окна ЭКГ (размер массива
AT).
            int cl            - Начальный код линии аннотации (если cl==-1, то
по умолчанию cl=44 - код "(t" ).
            int m_LT    - Максимальный размер массива LT.
            int m_uLT   - Максимальный размер массива uLT.
            int* L            - Текущая линия аннотации.
            int m_L           - Максимальный размер массива L.
```

```
Выход:      int* LT           - Уникальные линии аннотации (последовательность
кодов аннотации) хранящиеся в гнездовой структуре.
            int* uLT    - Указатели (курсоры) на начало гнезд в массиве LT.
            int* numL   - Число уникальных линий аннотации равно (размер массива
uLT равен numL+1).
Возвращает:  0 - Не найден начальный код линии аннотации.
                1 - Ошибок нет, линии аннотации выделены.
               -1 - Для массива LT требуется размер больше, чем m_LT.
               -2 - Для массива uLT требуется размер больше, чем m_uLT.
               -3 - Для массива L требуется размер больше, чем m_L.
Вызываемые функции: SeachL, AddL.
*/
ECG_C_API
int ALine(        int* AT,
                        int lenA,
                        int cl,
                        int* LT,
                        int* uLT,
                        int* numL,
                        int m_LT,
                        int m_uLT,
                        int* L,
                        int m_L
                  )
{
int i,c,r=0,lenL;
if (cl==-1) cl=44; //по умолчанию код "(t"
_m1:
            i=0;  uLT[0]=0;   *numL=0;    goto _m2;

_m2:
            if (i>=lenA)      { r=0; goto _m5;}
            if (AT[i]!=cl)    { i++; goto _m2; }
            if (AT[i]==cl)    { L[0]=cl; lenL=1; i++; goto _m3; }

_m3:
            if (i>=lenA)      { r=1; goto _m5;}
            if (lenL>=m_L)    { r=-3; goto _m5;}
            if (AT[i]!=cl)    {
                  L[lenL]=AT[i]; lenL++; i++; goto _m3; }
            if (AT[i]==cl)    {
                  c=SeachL(LT,uLT,numL,L,lenL,m_uLT,m_LT); goto _m4;  }

_m4:
            if (c<0)          { r=c; goto _m5;}
            if (c==0)         { goto _m2;  }
            if (c==1)         { AddL(LT,uLT,numL,L,lenL); goto _m2; }

_m5:
            return r;

}


/*
Функция: int SLine(     int* AT,
                              int* AS,
                              int lenA,
                              int* LT,
                              int* uLT,
                              int numL,
                              int  nL,
                              int* L,
```

```
                                int* LS,
                                int* lenLS,
                                int m_LS
                           )
Назначение: Нахождение номеров первых аннотированных точек линии с заданным
номером.
                    * Функция SLine реализована в автоматном стиле.
Вход: int* AT           - Коды аннотации в аннотированных точках окна ЭКГ.
            int* AS           - Номера аннотированных точках окна ЭКГ.
            int lenA    - Число аннотированных точек окна ЭКГ (размер массива AT
и AS).
            int* LT           - Уникальные линии аннотации (последовательность
кодов аннотации) хранящиеся в гнездовой структуре.
            int* uLT    - Указатели (курсоры) на начало гнезд в массиве LT.
            int numL    - Число уникальных линий аннотации равно (размер массива
uLT равен numL+1).
            int  nL           - Номер линии аннотации (0<=nl<numL numL+1 -
размер массива uLT).
            int* L            - Текущая линия аннотации.
            int m_LS    - Максимальный размер массива LS.
Выход:      int* LS           - Номера первых аннотированных точек линии с
номером nL.
            int* lenLS  - Размер массива LS.
Возвращает:  0 - Линия аннотации с номером nL не существует (массив LS -
пустой).
                 1 - Ошибок нет, номера первых точек линии аннотации с номером
nL выделены.
                -4 - Для массива LS требуется размер больше, чем m_LS.
*/
ECG_C_API
int SLine(        int* AT,
                        int* AS,
                        int lenA,
                        int* LT,
                        int* uLT,
                        int numL,
                        int  nL,
                        int* L,
                        int* LS,
                        int* lenLS,
                        int m_LS
                  )
{
int i,j,k=0,cl,r=0,lenL;

_m1:  if (nL>=numL) { r=0; goto _m7;}
            lenL=uLT[nL+1]-uLT[nL]; i=uLT[nL]; cl=LT[i]; j=0;  k=0;     goto _m2;

_m2:
            if (j>=lenL)      { i=0; goto _m3;}
            L[j]=LT[i]; i++; j++; goto _m2;
_m3:
            if (i>=lenA)      { r=1; goto _m7;}
            if (AT[i]!=cl)    { i++; goto _m3;}
            if (AT[i]==cl)    { j=0; goto _m4;}

_m4:
            if ((i>=lenA))    { r=1; goto _m7;}
            if (j>=lenL)      goto _m5;
            if (AT[i]==L[j]){ i++; j++; goto _m4; }
            if (AT[i]!=L[j]){ goto _m3; }
_m5:
            if (AT[i]==cl)    {
```

```
            LS[k]=AS[i-lenL]; k++; goto _m6;
            }
            goto _m3;
_m6:
            if (k>=m_LS)      { r=-4; goto _m7;}
            goto _m3;

_m7:
            *lenLS=k;
            return r;

}
```

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

«___» _____ _____ г.

_____ Берест Д.А.

(подпись) *(Ф. И. О.)*