на тему: «Разработка информационно-поисковой системы предприятия»

Орел 2020

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc42808722)

[1 ПОСТАНОВКА ЗАДАЧИ И РАЗРАБОТКА ТРЕБОВАНИЙ 5](#_Toc42808723)

[2 ОБОСНОВАНИЕ ВЫБОРА МЕТОДА РЕШЕНИЯ И СТРУКТУР ДАННЫХ 7](#_Toc42808724)

[3 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ 10](#_Toc42808725)

[4 ОСОБЕННОСТИ ПРОГРАММНОЙ РЕАЛИЗИЦИИ 22](#_Toc42808726)

[5 ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА 24](#_Toc42808727)

[ЗАКЛЮЧЕНИЕ 30](#_Toc42808728)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 31](#_Toc42808729)

[ПРИЛОЖЕНИЕ А – ЛИСТИНГ ФАЙЛА “mainwindow.cpp” 32](#_Toc42808730)

[ПРИЛОЖЕНИЕ B – ЛИСТИНГ ФАЙЛА “mainwindow.h” 48](#_Toc42808733)

[ПРИЛОЖЕНИЕ C – ЛИСТИНГ ФАЙЛА «addaqua.cpp» 49](#_Toc42808736)

[ПРИЛОЖЕНИЕ D – ЛИСТИНГ ФАЙЛА “addaqua.h” 53](#_Toc42808739)

[ПРИЛОЖЕНИЕ E – ЛИСТИНГ ФАЙЛА “addcardio.cpp” 54](#_Toc42808742)

[ПРИЛОЖЕНИЕ F – ЛИСТИНГ ФАЙЛА “addcardio.h” 58](#_Toc42808745)

[ПРИЛОЖЕНИЕ G – ЛИСТИНГ ФАЙЛА “addpower.cpp” 59](#_Toc42808748)

[ПРИЛОЖЕНИЕ H – ЛИСТИНГ ФАЙЛА “addpower.h” 63](#_Toc42808751)

[ПРИЛОЖЕНИЕ I – ЛИСТИНГ ФАЙЛА “client1.cpp” 64](#_Toc42808754)

[ПРИЛОЖЕНИЕ J – ЛИСТИНГ ФАЙЛА “client1.h” 66](#_Toc42808757)

[ПРИЛОЖЕНИЕ K – ЛИСТИНГ ФАЙЛА “show\_all.cpp” 68](#_Toc42808760)

[ПРИЛОЖЕНИЕ L – ЛИСТИНГ ФАЙЛА “show\_all.h” 69](#_Toc42808763)

# ВВЕДЕНИЕ

Современный этап развития цивилизации характеризуется переходом от индустриального общества к информационному. Все это выражается в процессе создания различных видов систем, как встроенных в уникальные объекты информационно-технологических комплексов. В последнее время они стали неотъемлемой частью нашей жизни. Проблема поиска и сбора информации является одной из важнейших проблем информационно-поисковых систем.

Целью проекта является разработка информационно-поисковой системы «Фитнес-клуб» на языке С++ с использованием объектно-ориентированного программирования.

Задачами проекта, которые необходимо выполнить для достижения поставленной цели, являются:

1. Выбор метода решения поставленной задачи.
2. Выбор подходящих структур данных.
3. Разработка алгоритмов решения.
4. Реализация программы на языке C++.
5. Описание пользовательского интерфейса.
6. ПОСТАНОВКА ЗАДАЧИ И РАЗРАБОТКА ТРЕБОВАНИЙ

Целью проекта является разработка информационно-поисковой системы «Фитнес-клуб» на языке С++ с использованием объектно-ориентированного программирования.

В программе разработать родительский класс и три дочерних класса (с соблюдением инкапсуляции). Дочерние классы дополнить новыми полями и методами.

Для хранения информации об объектах (родительского и дочерних классов) разработать одну динамическую структуру данных (нельзя использовать STL; примеры динамических структур данных: массив, список, стек, очередь).

Программа должна иметь удобный графический интерфейс с обязательными элементами:

1) таблица (служить только для отображения информации, которая содержится в динамической структуре данных);

2) главное меню с пунктами «Файл» (подпункты «Создать...», «Открыть...», «Сохранить», «Сохранить как...», «Выход») и «Справка» (подпункты «Руководство», «О программе»). Все подпункты главного меню необходимо реализовать традиционно (как должны работать подпункты меню можно посмотреть, например, в программах Microsoft Office). В программе определить файл. Пункт главного меню «Файл» должен работать с файлом и динамической структурой данных.

Основной функционал программы:

1) добавление, удаление и редактирование информации;

2) поиск информации по заданным критериям (не менее трех критериев поиска);

3) отображение всей информации и информации об объектах выбранного класса из динамической структуры данных в табличном виде;

4) реализация главного меню.

Основной функционал реализовывать с соблюдением принципа полиморфизма.

При разработке программы продемонстрировать понимание основных принципов объектно-ориентированного программирования (инкапсуляция, наследование, полиморфизм).

На основе всего вышесказанного были сформулированы функциональные требования:

1. Разработать один родительский и три дочерних класса.
2. Использовать динамическую структуру как альтернативу STL.
3. Разработать удобный графический интерфейс.
4. Реализовать основной функционал программы.
5. Реализовать программу на языке С++ с использованием объектно-ориентированного программирования.

Минимальные системные требования:

1. ОС: 64-разрядная версия Windows 10.
2. Процессор: Intel Core i5 3570 или AMD FX-6350.
3. ОЗУ: 8 Гб
4. Видеокарта: NVIDIA GTX 760, AMD Radeon 7970 / R9280X.
5. Память видеокарты: 2 Гб
6. Жёсткий диск: не менее 50 Гб свободного места
7. ОБОСНОВАНИЕ ВЫБОРА МЕТОДА РЕШЕНИЯ И СТРУКТУР ДАННЫХ

Структуры в С++ используются для логического и физического объединения данных произвольных типов, так же как массивы служат для группирования данных одного типа.

Для решения поставленной задачи следует использовать динамическую структуру данных.

Динамические структуры данных - это любая структура данных, занимаемый объем памяти которой не является фиксированным. Иными словами, в подобной структуре может храниться как два, пять, двадцать элементов, так и ничего [5].

Выделяют следующие динамические структуры данных: список, очередь, стек и массив.

Для реализации динамической структуры данных будет использоваться массив. Так как массив - это группа однотипных элементов, которые упорядочены по индексам. Его можно использовать для хранения информации. В массиве легко вычислить адрес элемента (по индексу), одинаковое время доступа ко всем элементам и элементы имеют малый размер (они состоят только из информационного поля).

В работе для решения поставленной задачи будет использоваться динамический массив. Динамический массив - это массив, у которого количество ячеек можно задавать и переменной, и числовой константой [3]. Его размер заранее не фиксирован и может изменяться во время исполнения программы. Возможность изменения размера отличает динамический массив от статического, размер которого задаётся на момент компиляции программы. Для изменения размера динамического массива язык программирования, поддерживающий такие массивы, должен предоставлять встроенную функцию или оператор. Динамические массивы дают возможность более гибкой работы с данными, так как позволяют не прогнозировать хранимые объёмы данных, а регулировать размер массива в соответствии с реально необходимыми объёмами [8].

На основе приведенных выше выводов, учитывая особенности функциональных требований, был сделан выбор в пользу динамического массива для упрощения читабельности кода и ввиду того, что в массиве упрощается обращение к элементам массива (по индексу), что необходимо для выполнения операций добавления, поиска, вставки и удаления. Ведь в отличие от массива, в стеке можно перемещаться только в одном направлении, что сильно затрудняет поиск нужного элемента. А в списке вся память соединена в одну цепочку, и чтобы обратиться к необходимому элементу, надо пройти через все элементы, стоящие до него.

При добавлении информации в массив можно использовать следующие способы:

1. Добавить новый объект в определенную позицию.
2. Добавить новый объект в начало.
3. Добавить новый объект в конец.

Для добавления в программе будет использоваться добавление элемента в конец, поскольку необходимо перемещать каждый объект после точки вставки. Каждый элемент после точки вставки должен быть скопирован на соседнее место во внутреннем массиве, и после завершения копирования пустота заполняется новым объектом. Так как необходимо перемещать каждый объект после точки вставки, то наилучшим случаем будет добавление элемента к концу [9].

Оно также эффективнее по времени и памяти в отличие от добавления в начало или в указанную позицию.

При поиске определенного значения элемента можно использовать последовательный, индексно-последовательный или двоичный поиски. В программе будет использоваться последовательный поиск, так как он прост в реализации, не требует сортировки значений множества, дополнительной памяти и дополнительного анализа функций.

При изменении элемента, будет считываться номер строки таблицы, который записывается в текстовом поле. Сначала изменяются данные в самой таблице. Затем, при нажатии кнопки «Главное меню», все элементы будут перезаписываться.

При удалении объекта сначала удаляются данные из таблицы. Потом происходит смещение элементов массива по количеству удаленных строк в таблице.

Файл является удобной структурой данных для постоянного хранения информации. Благодаря ему данные не будут утеряны после завершения работы, и будут снова загружены из файла для дальнейшей работы при следующем запуске программы. В программе не будет использоваться бинарный файл, так как нам потребуются дополнительные преобразования из строки в массив символов и контроль размера всех читаемых массивов символов. Поэтому будет использован текстовой файл, так как в нем есть возможность построчного чтения и записи в понятном для пользователя виде.

Представление программы будет выполнено в виде графического интерфейса для большего удобства пользователя. Не используется консоль, так как консоль не позволяет обеспечить вывод данных в таблице и удобное редактирование выбранной строки.

Таким образом, была выбрана интегрированная среда разработки Qt Creator.

1. ПРОЕКТИРОВАНИЕ ПРОГРАММЫ

В программе будет содержаться шесть модулей, включающие основную и вспомогательные функции.

На рисунке 1 представлена схема модулей программы.

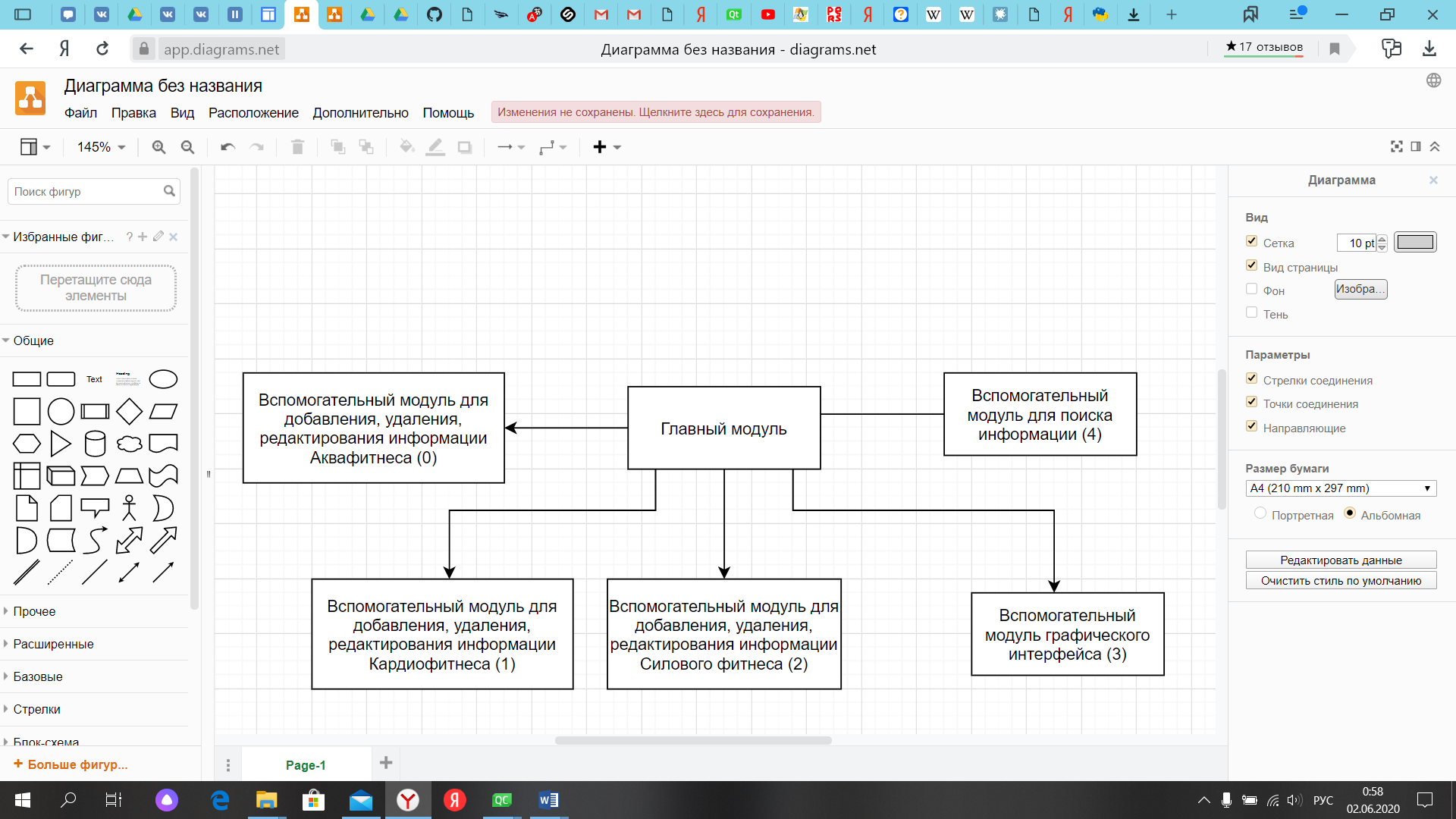


Рисунок 1 – Схема модулей программы

Главный модуль – модуль главного окна, содержащий методы для обработки кнопок, выхода из программы, отображения информации для руководства и сведений о программе. Также он включает в себя родительский класс, три дочерних класса с полями и методами, динамический массив объектов классов и функции для вызова остальных вспомогательных модулей.

Вспомогательный модуль (0) – модуль, в котором считываются введенные данные для записи на «Аквафитнес» и происходит преобразование их в конкретный тип, добавление преобразованных данных в массив. Также в нем происходит удаление выбранного элемента из массива, изменение выбранного элемента в массиве и смещение остальных элементов для освобождения памяти.

Вспомогательный модуль (1) – модуль, в котором считываются введенные данные для записи на «Кардиофитнес» и происходит преобразование их в конкретный тип, добавление преобразованных данных в массив. Также в нем происходит удаление выбранного элемента из массива, изменение выбранного элемента в массиве и смещение остальных элементов для освобождения памяти.

Вспомогательный модуль (2) – модуль, в котором считываются введенные данные для записи на «Силовой фитнес» и происходит преобразование их в конкретный тип, добавление преобразованных данных в массив. Также в нем происходит удаление выбранного элемента из массива, изменение выбранного элемента в массиве и смещение остальных элементов для освобождения памяти.

Вспомогательный модуль (3) – модуль поиска, в котором считываются данные, затем происходит поиск конкретной информации из массива и передача этой информации в таблицу.

Вспомогательный модуль (4) - модуль главного окна, который представляет собой графическое отображение программы.

На рисунке 2 представлена диаграмма классов. В программе используется один родительский класс и три дочерних. Я выбрала такие методы, потому что у нас все поля классов приватные и мы можем обратиться к ним только через эти методы, для того чтоб достать информацию из массива и передать её в таблицу.

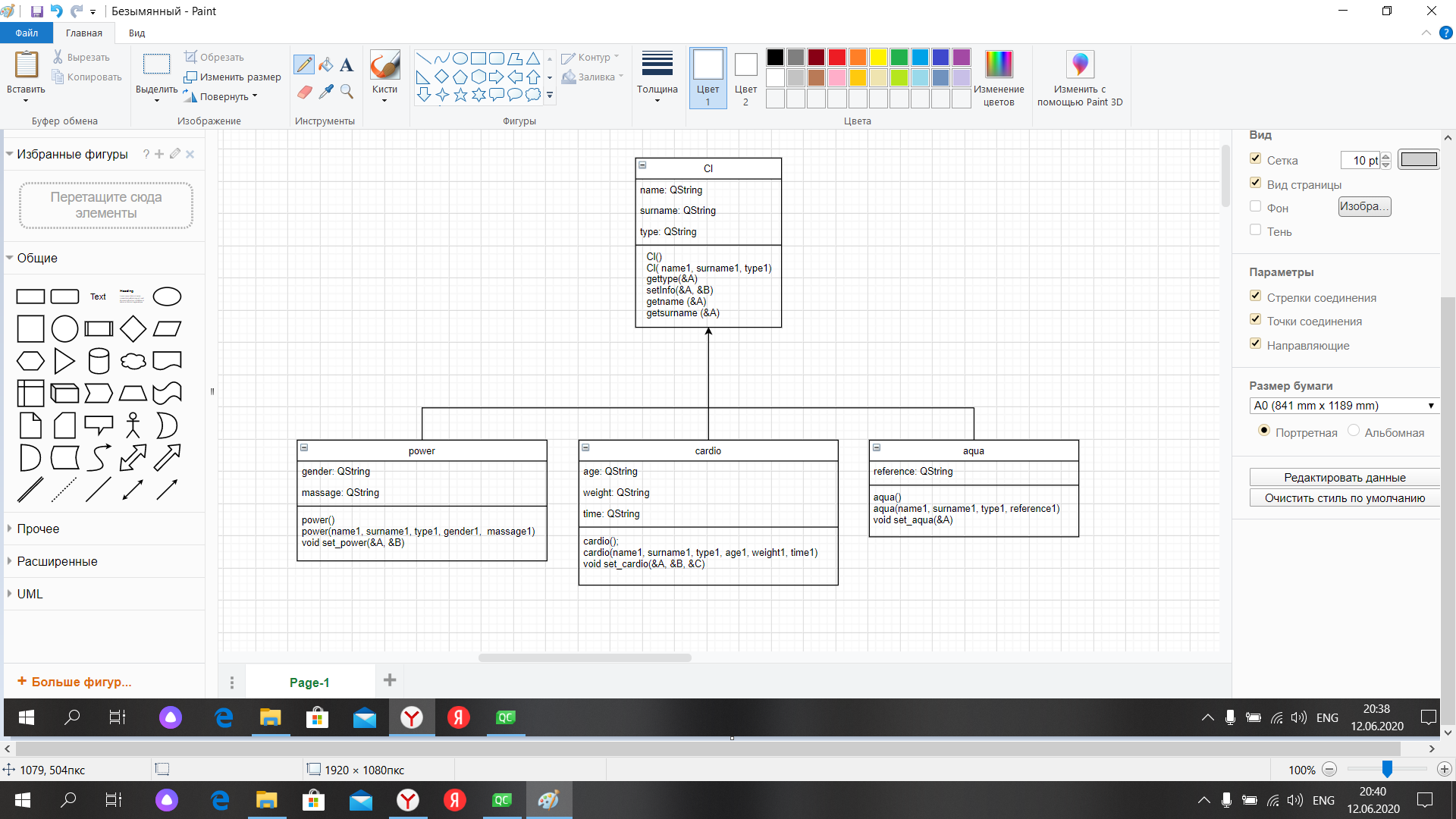


Рисунок 2 – Диаграмма классов

На рисунке 3 представлена схема алгоритмов главного модуля.

Алгоритм нажатия кнопок в главном модуле имеет следующий вид:

1. Нажат подпункт меню «Выход». Если нет, то переходим на шаг 2, иначе на шаг 19.
2. Нажата кнопка «Силовой фитнес», если да переходим на шаг 16.
3. Нажата кнопка «Кардиофитнес», если да переходим на шаг 16.
4. Нажата кнопка «Аквафитнес», если да переходим на шаг 16.
5. Нажата кнопка «Отобразить все», если да переходим на шаг 16.
6. Нажата кнопка «Поиск по», если да переходим на шаг 16.
7. Нажат пункт меню «Файл», если нет, переходим на шаг 12.
8. Нажат подпункт меню «Открыть», если да переходим на шаг

16.

1. Нажат подпункт меню «Сохранить», если да переходим на шаг

16.

1. Нажат подпункт меню «Сохранить как…», если да переходим

на шаг 16.

1. Нажат подпункт меню «Создать новый проект», если да, переходим на шаг 16.
2. Нажат пункт меню «Справка», если да, переходим на шаг 15.
3. Нажат подпункт меню «Руководство», если да, переходим на

шаг 15.

1. Нажат подпункт меню «О программе», если да, переходим на шаг 15.
2. Вывести на экран необходимую информацию.
3. Открыть соответствующее окно.
4. Перейти на шаг 1.
5. Завершение программы.

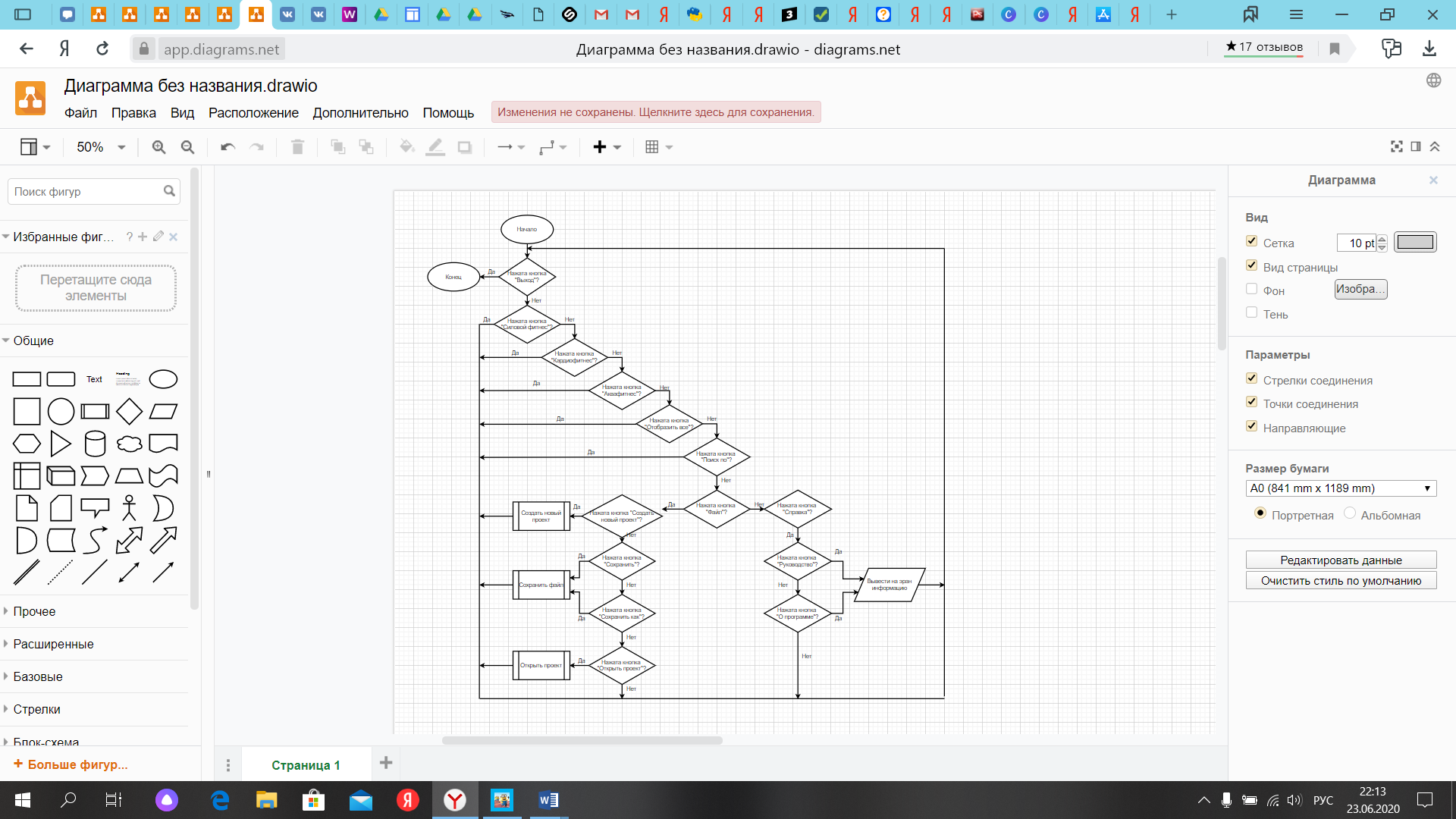


Рисунок 3 – Схема главного модуля

На рисунке 4 представлена схема алгоритма добавления клиентов.

* 1. Нажата кнопка «Добавить».
  2. Введены все данные, если да, то переходим на шаг 4.
  3. Вывести сообщение об ошибке и перейти на шаг 11.
  4. Такие данные уже есть, если да, то переходим на шаг 11.
  5. Получаем введенные данные.
  6. Увеличиваем массив.
  7. Увеличиваем количество определенного фитнеса.
  8. Присваиваем нужный тип.
  9. Создаем объект выбранного дочернего класса.
  10. Передаем полученные данные в конструктор дочернего класса.
  11. Присваиваем последнему элементу массива созданный объект.
  12. Отображаем данные в таблице.
  13. Завершение алгоритма.

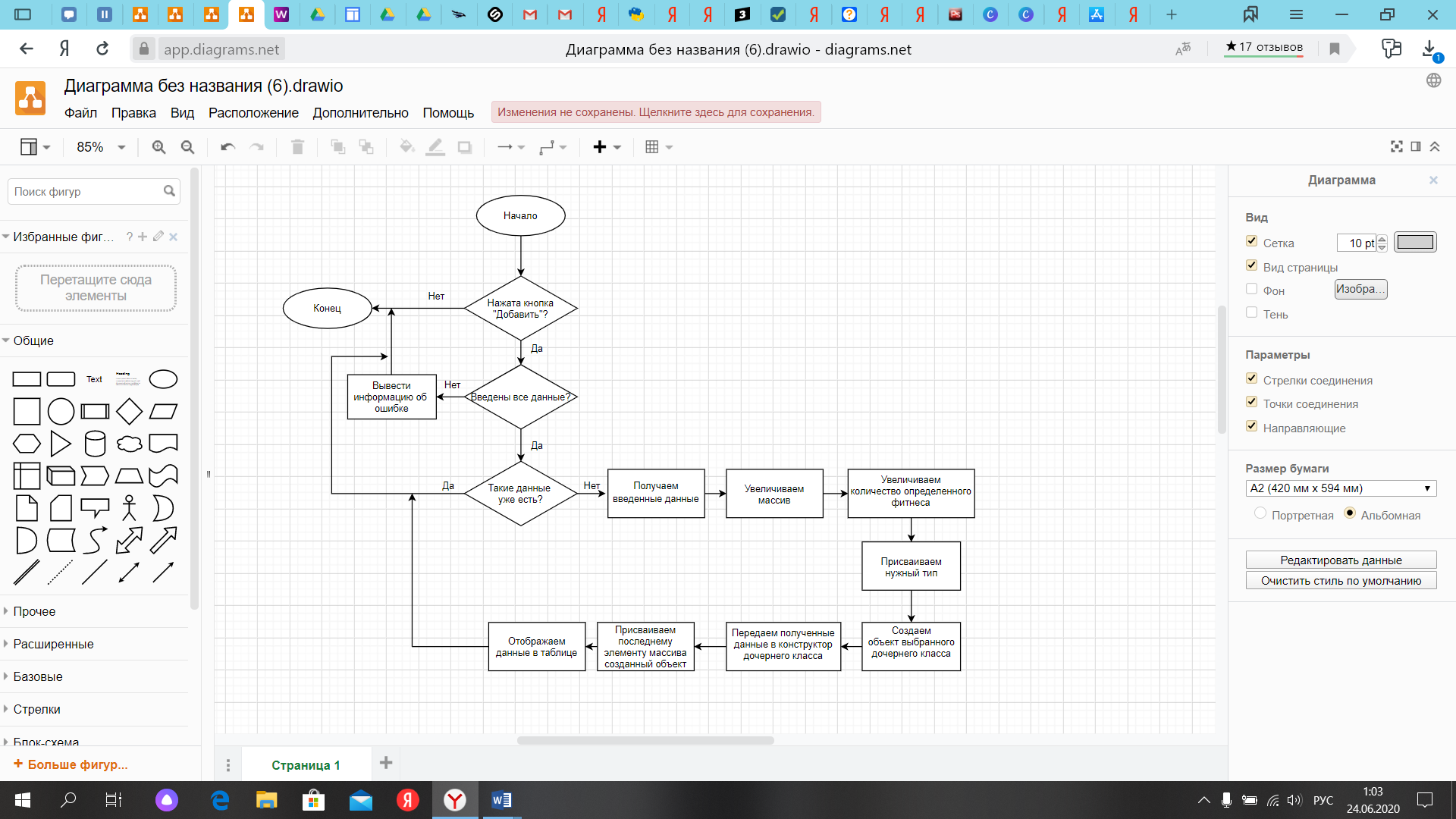


Рисунок 4 – Схема добавления клиента

На рисунке 5 представлена схема алгоритма удаления клиентов.

1. Нажата кнопка «Удалить».
2. Вводим номер строки удаления.
3. Если удалены все элементы, то переходим на шаг 5
4. Если удалены не все, то переходим на шаг 8
5. Находим выбранный элемент.
6. Смещаем все элементы.
7. Удаляем последний элемент из таблицы.
8. Удалено нужное количество, если нет, то на шаг 5.
9. Нажата кнопка «Главное меню».
10. Считываем данные из таблицы.
11. Передаем новые данные в конструктор дочернего класса.
12. Присваиваем нужному элементу массива новые данные
13. Завершение алгоритма.

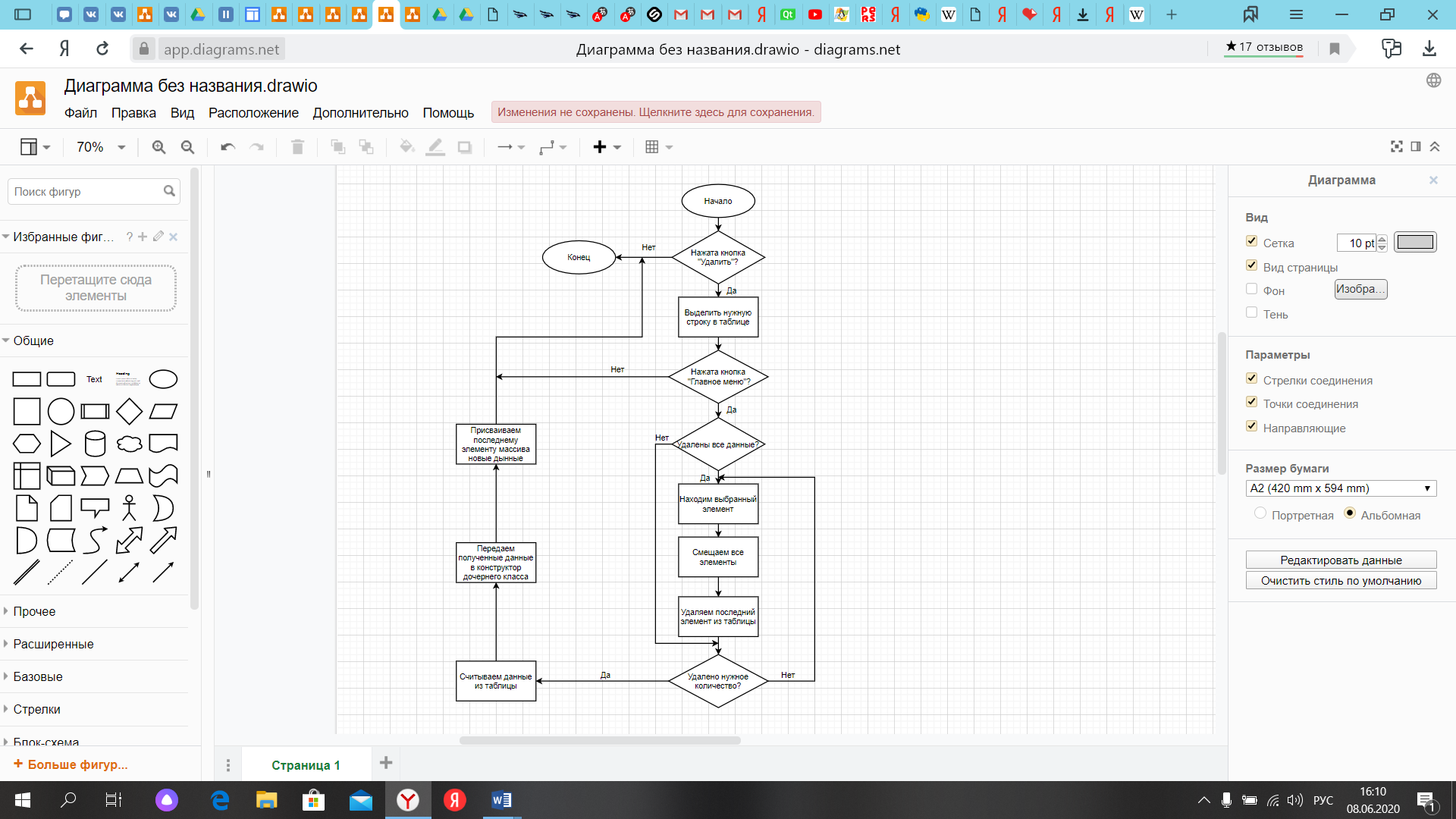


Рисунок 5 – Схема удаления клиента

На рисунке 6 представлена схема алгоритма редактирования клиентов.

1. Нажата кнопка «Редактировать».
2. Выделить нужную строку в таблице.
3. Введены все данные, если да, то переходим на шаг 5.
4. Вывести сообщение об ошибке и перейти на шаг 11.
5. Считываем введенные данные в таблицу.
6. Отображение информации в таблице.
7. Нажата кнопка «Главное меню».
8. Считываем данные из таблицы.
9. Передаем новые данные в конструктор дочернего класса.
10. Присваиваем нужному элементу массива новые данные
11. Завершение алгоритма

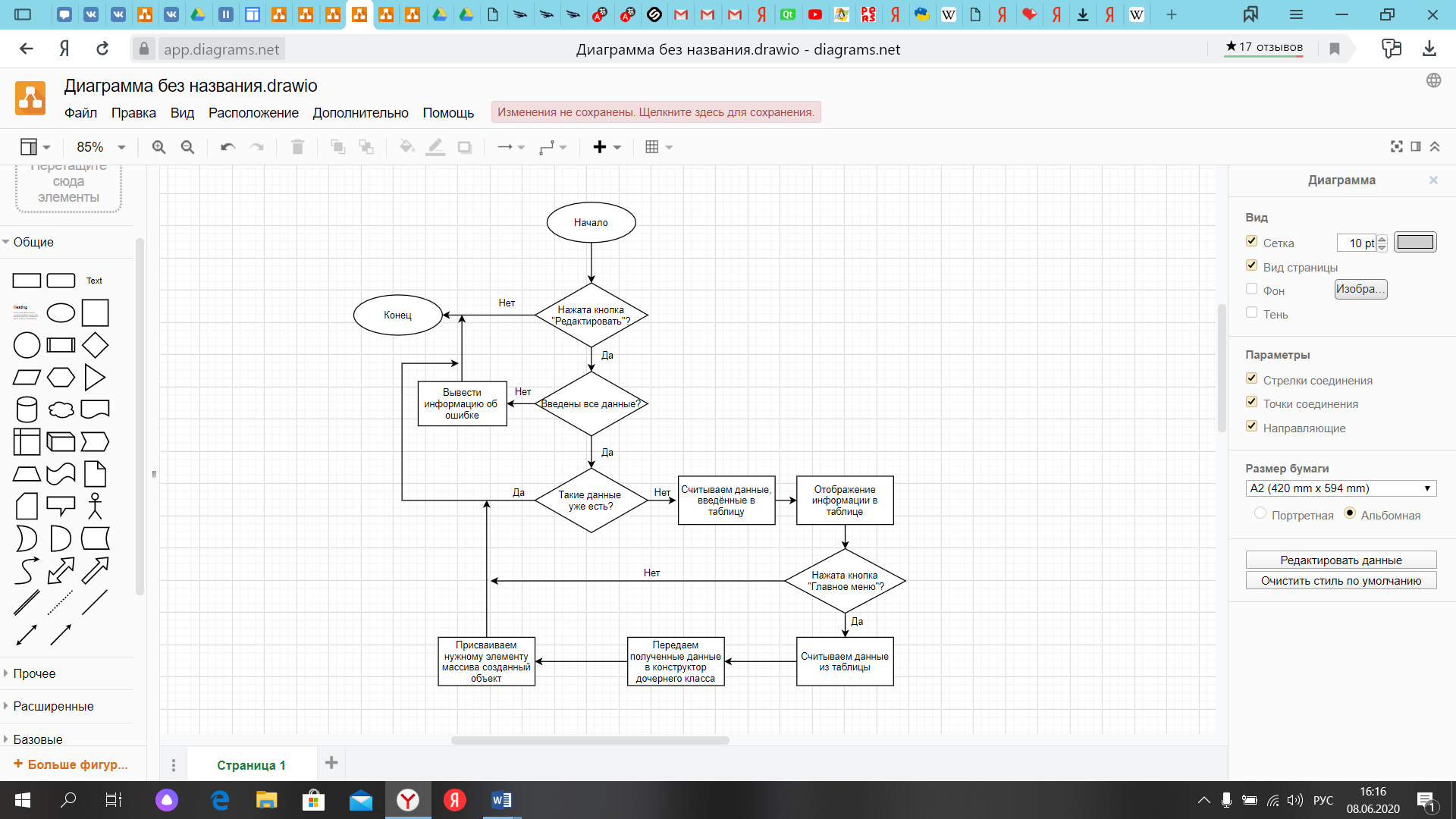


Рисунок 6 – Схема редактирования информации о клиенте

На рисунке 7 представлена схема алгоритма отображения всех клиентов.

1. Нажата кнопка «Показать все».
2. Считываем все данные из массива.
3. Заполняем таблицу информацией.
4. Отображение информации в таблице.
5. Завершение алгоритма.

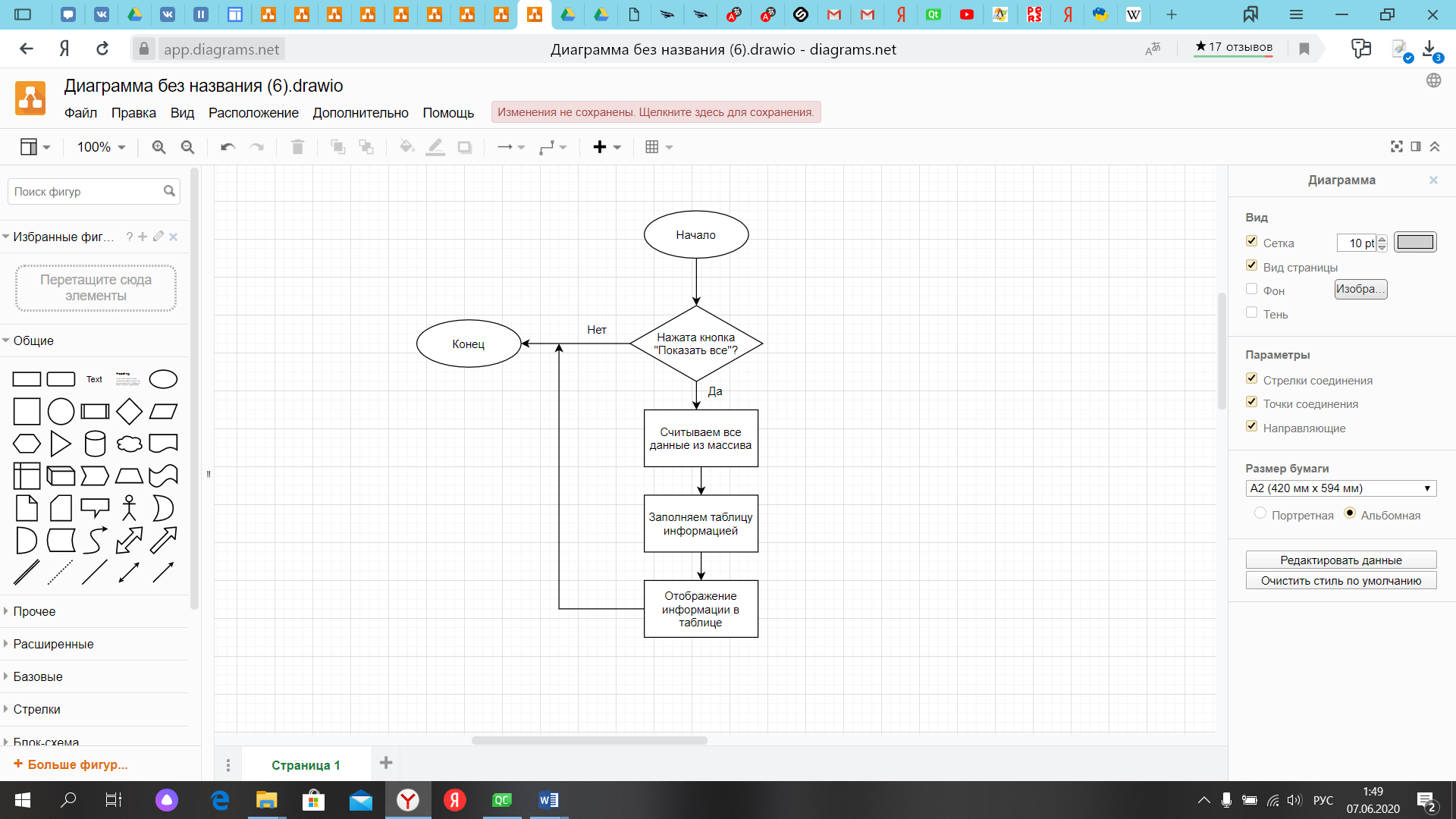


Рисунок 7 – Схема отображения клиентов

На рисунке 8 представлена схема алгоритма поиска клиентов.

1. Выбран поиск по имени, если да перейти на шаг 4.
2. Выбран поиск по фамилии, если да перейти на шаг 4.
3. Выбран поиск по номеру в таблице, если нет, перейти на шаг 14.
4. Поиск нужной информации в массиве.
5. Информация найдена, если нет, то на перейти на шаг 14.
6. Считать тип.
7. Тип «cardio», если да, то перейти на шаг 10.
8. Тип «aqua», если да, то перейти на шаг 11.
9. Тип «power», если да, то на шаг 12.
10. Отправить информацию об объекте в таблицу о кардиофитнесе
11. Отправить информацию об объекте в таблицу об аквафитнесе.
12. Отправить информацию об объекте в таблицу о силовом фитнесе.
13. Отобразить информацию в таблице.
14. Завершение алгоритма.

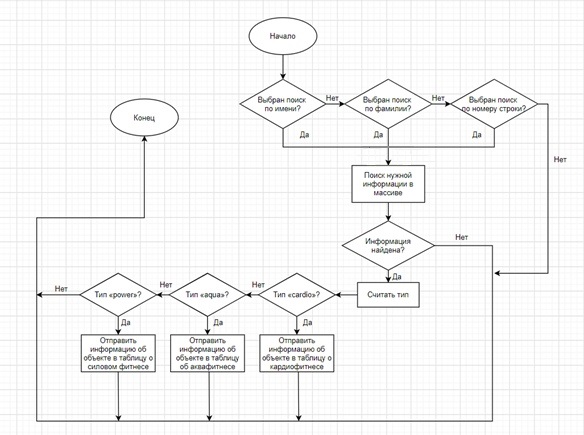


Рисунок 8 – Схема поиска клиентов

На рисунке 9 представлена схема сохранения данных в файл.

1. Нажата кнопка «Сохранить», если нет, то перейти на шаг 9.
2. Кнопка «Открыть» была нажата, если да, то перейти на шаг 5.
3. Кнопка «Сохранить как…» была нажата, если да, то перейти на шаг 5.
4. Получить путь к файлу.
5. Считать все элементы из массива.
6. Записать каждый элемент в файл по полученному пути. Перейти на шаг 9.
7. Считать все элементы из массива.
8. Записать каждый элемент в открытый файл.
9. Завершение алгоритма.

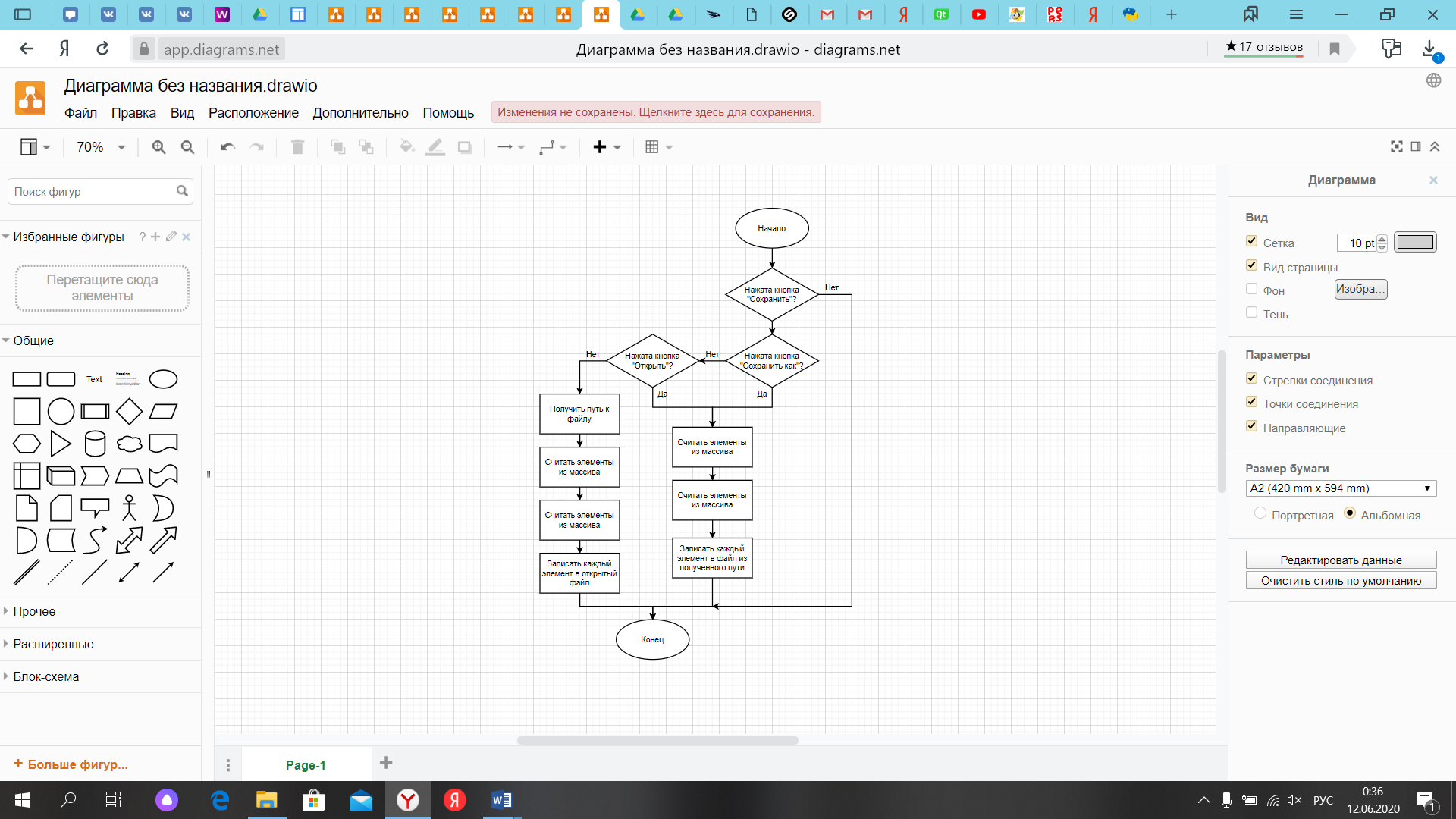


Рисунок 9 – Схема сохранения данных в файл

На рисунке 10 представлена схема открытия файла.

1. Нажать кнопка «Открыть», если нет, то перейти на шаг.
2. Получить путь к файлу.
3. Считывать построчно данные из файла.
4. Считать тип из файла.
5. Увеличить массив на один элемент.
6. Увеличить количество конкретного товара на один.
7. Создать объект нужного класса.
8. Передать полученные данные в конструктор дочернего класса.
9. Присвоить последнему элементу массива новый объект.
10. Завершение алгоритма.

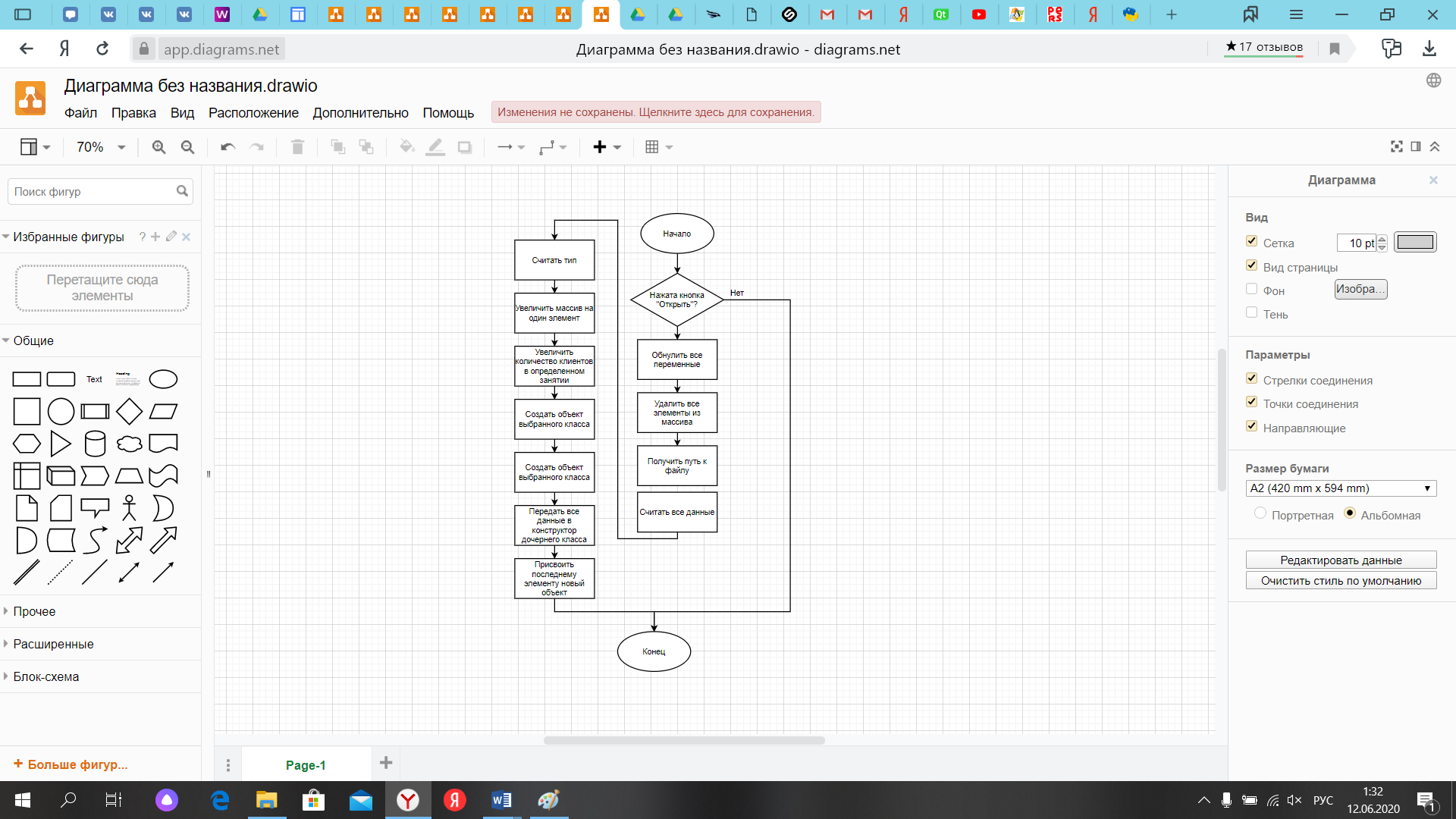


Рисунок 10 – Схема открытия файла

На рисунке 11 представлена схема сохранения файла как.

1. Нажата кнопка «Сохранить как…», если нет, то перейти на шаг 6.
2. Вывести сообщение об ошибке и перейти на шаг.
3. Получить путь к файлу.
4. Считать все данные из массива.
5. Записать построчно каждый элемент в файл.
6. Завершение алгоритма.

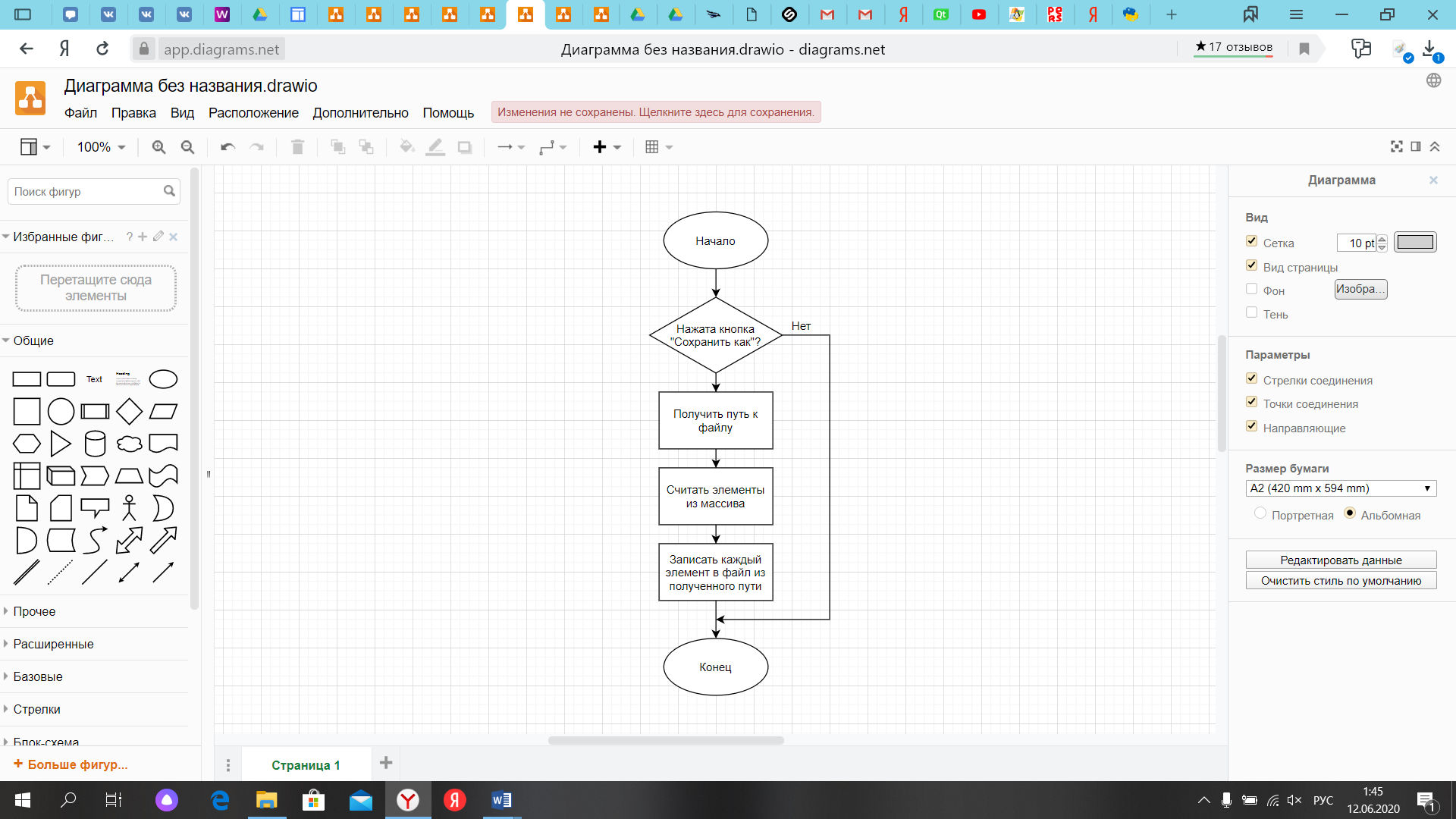


Рисунок 11 – Схема открытия файла

1. ОСОБЕННОСТИ ПРОГРАММНОЙ РЕАЛИЗИЦИИ

Главной структурой в программе, основанной на объектно-ориентированной методологии, является класс.

В силу того, что необходимо реализовать удобный графический интерфейс, то в программе присутствуют 5 форм, один родительский и три дочерних класса для логической работы информационно-поисковой системы. Форма «mainwindow» отвечает за работу главного окна, за добавление, редактирование и удаление клиентов, «addaqua» - за запись информации о клиентах в таблицу аквафитнеса, «addcardio» - за запись информации о клиентах в таблицу кардиофитнеса, «addpower» - за запись информации щ клиентах в таблицу силового фитнеса, «show\_all» - за отображение всей информации.

Класс «Cl» содержит 3 приватных поля типа «QString», а именно: «type» (тип фитнеса), «name» (имя клиента), «surname» (фамилия клиента). В данном классе имеются следующие публичные методы:

1. Конструктор класса «Сl» создает и инициализирует все поля нашего объекта.
2. Функция «gettype» типа «void», который извлекает тип из класса.
3. Функция «setInfo» типа «void», который извлекает имя и фамилию из класса.
4. Функция «getname» типа «void», который извлекает имя из класса.
5. Функция «getsurname» типа «void», который извлекает фамилию из класса.

Класс «power» является дочерним классом класса «Cl» содержит 2 приватных поля типа «QString», а именно: «gender» (пол клиента), «massage» (массаж). В данном классе имеются следующие публичные методы:

1. «power» - конструктор с параметрами для записи данных в класс, внутри которого находится конструктор «Cl» для записи части данных в родительский класс.
2. Функция «set\_power» типа «void», который извлекает пол клиента и необходимость массажа из класса.

Класс «cardio» является дочерним классом класса «Cl» содержит 2 приватных поля типа «QString», а именно: «age» (возраст клиента), «weight» (вес клиента), «time» (количество часов тренировок в неделю). В данном классе имеются следующие публичные методы:

1. «cardio» - конструктор с параметрами для записи данных в класс, внутри которого находится конструктор «Cl» для записи части данных в родительский класс.
2. Функция «set\_cardio» типа «void», который извлекает возраст клиента, его вес и количество часов тренировок из класса.

Класс «aqua» является дочерним классом класса «Cl» содержит 1 приватное поле типа «QString», а именно: «reference» (наличие справки на разрешение посещения бассейна). В данном классе имеются следующие публичные методы:

1. «aqua» - конструктор с параметрами для записи данных в класс, внутри которого находится конструктор «Cl» для записи части данных в родительский класс.
2. Функция «set\_aqua» типа «void», который извлекает наличие справки из класса.
3. ОПИСАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

При запуске программы пользователь видит перед собой главное окно, представленное на рисунке 12.

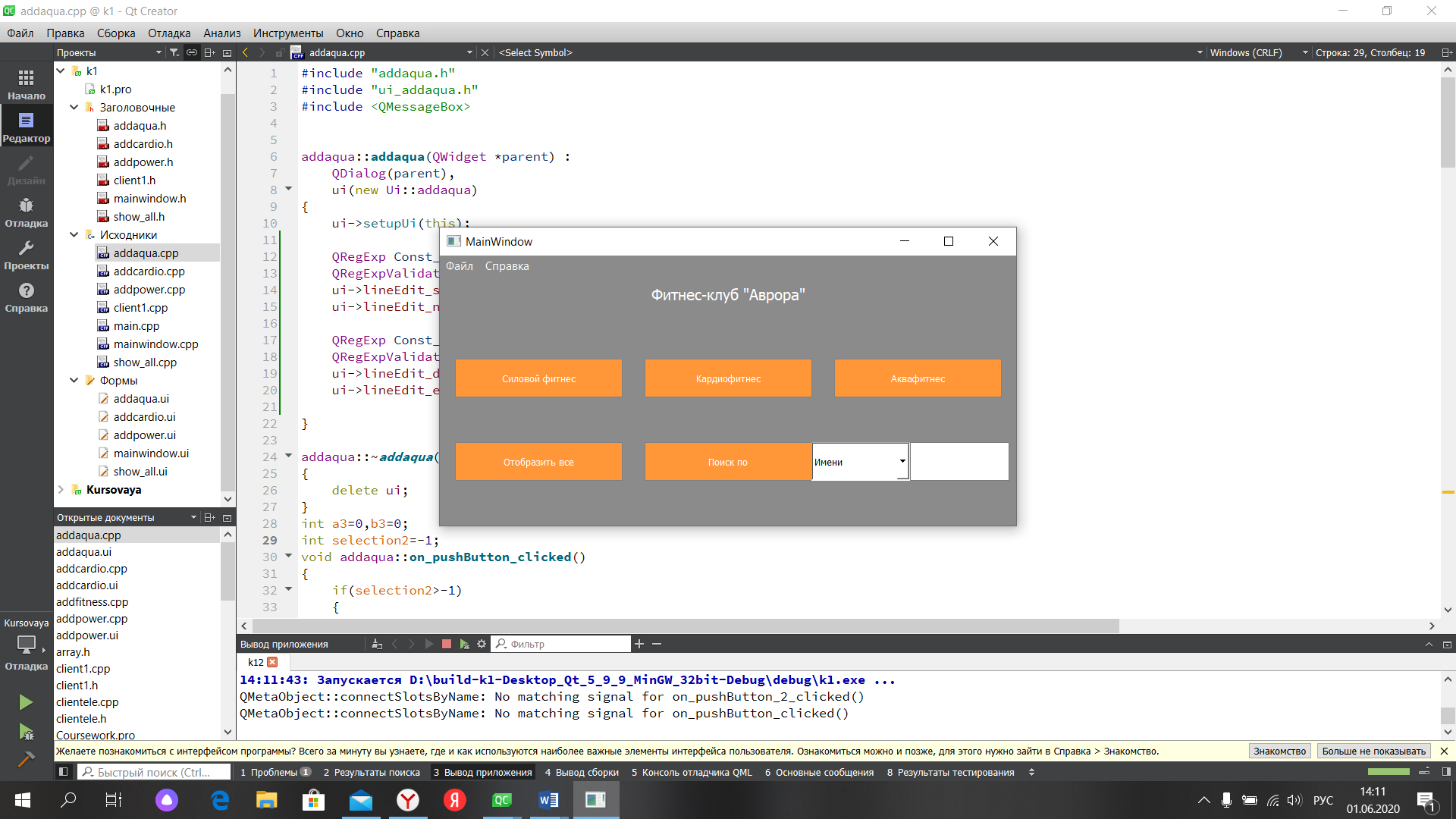


Рисунок 12 – Главное окно программы

После нажатия на соответствующую кнопку, пользователь перейдет в новое окно (в зависимости от того, куда он хочет записаться). Кнопка «Создать новый проект», появляющаяся после нажатия на кнопку «Файл», не имеет своего окна.

В программе действуют ограничения на текстовые поля. Таким образом, в поля «Имя» и «Фамилия» принимаются только символы, с любым регистром букв. В поля для ввода возраста и веса, индексов удаления или изменения принимает на ввод только числа.



Рисунок 13 – Окно записи на Силовой фитнес

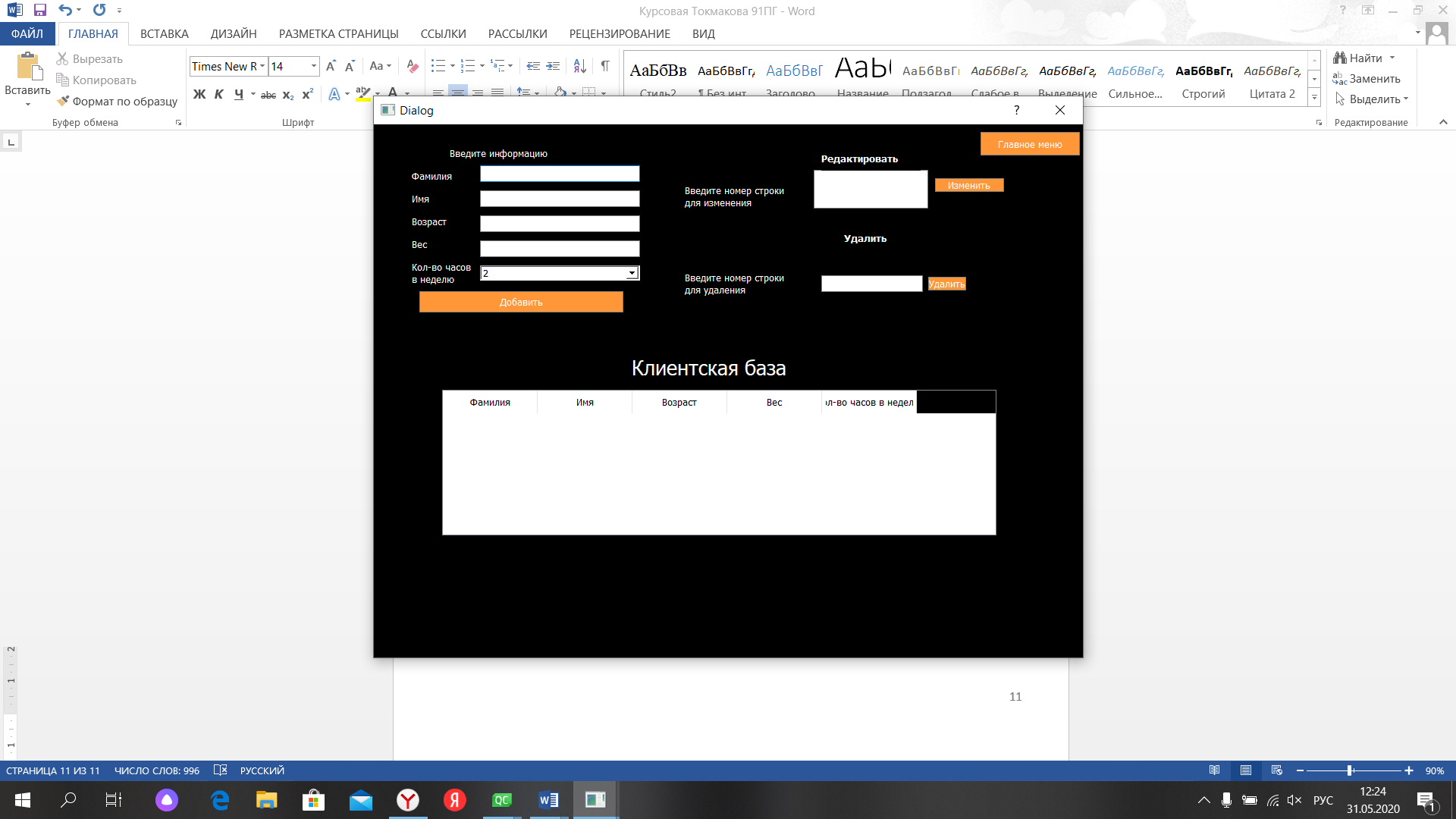


Рисунок 14 – Окно записи на Кардиофитнес

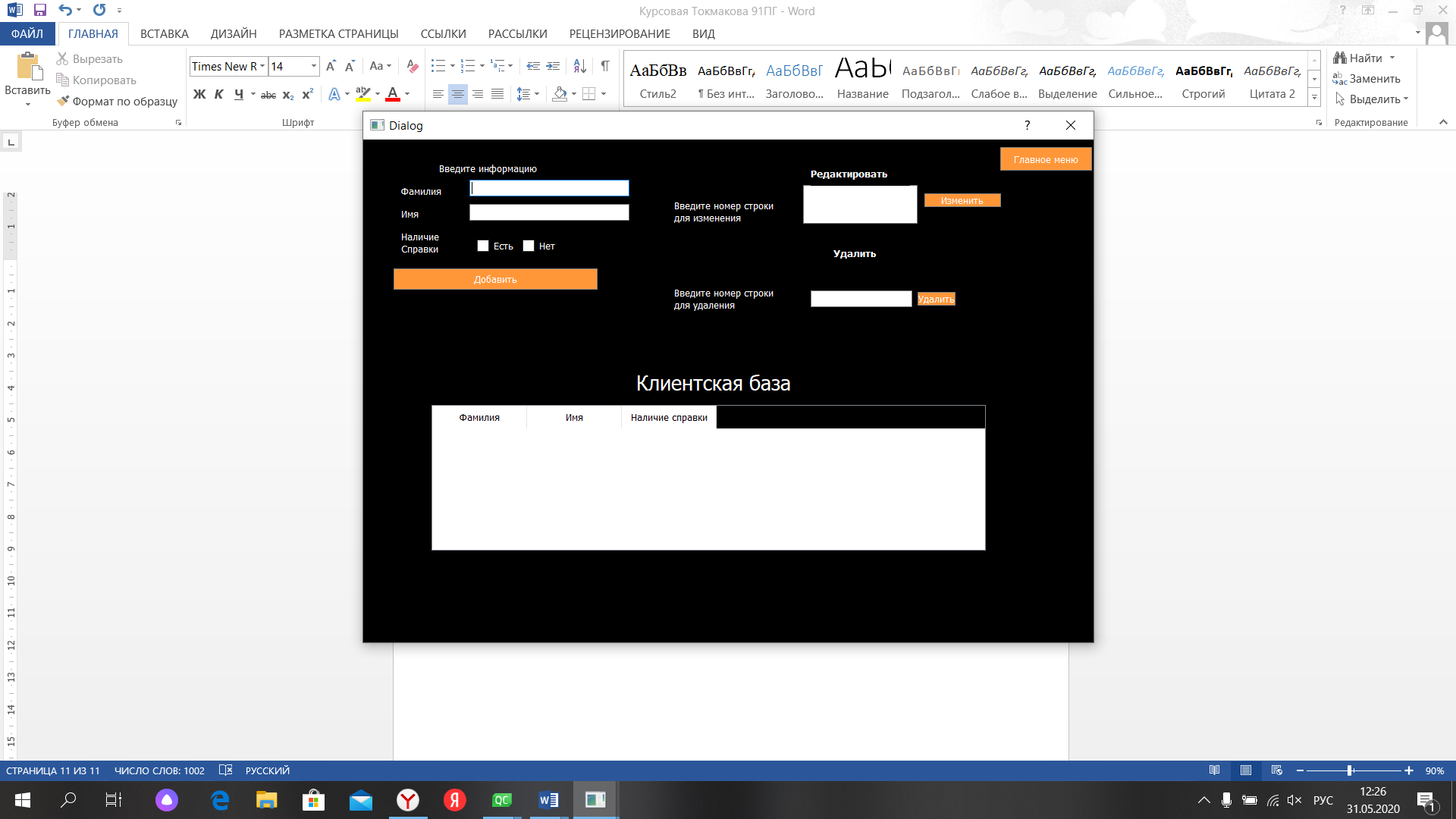


Рисунок 15 – Окно записи на Аквафитнес

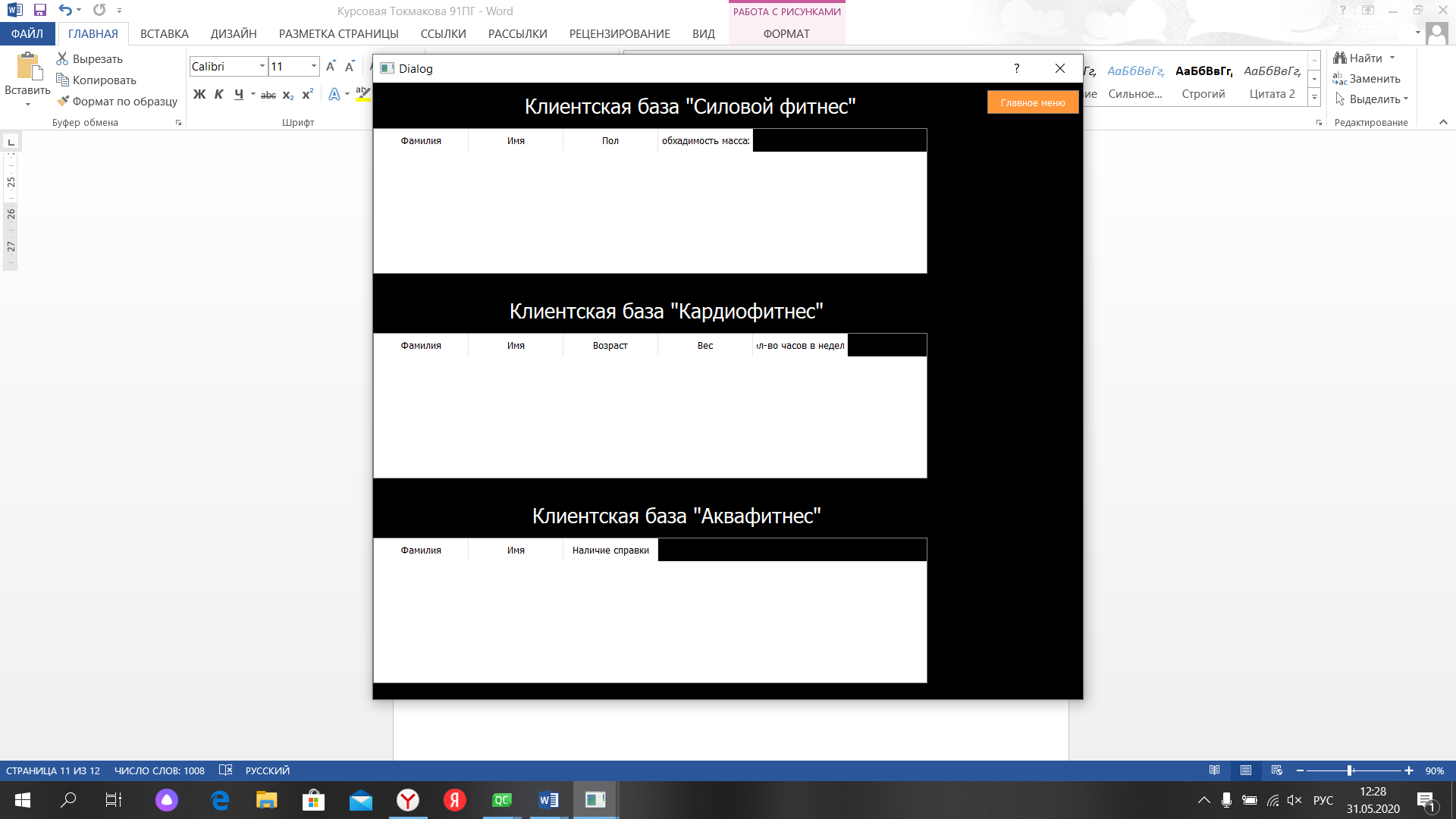


Рисунок 16 – Окно отображения всей информации

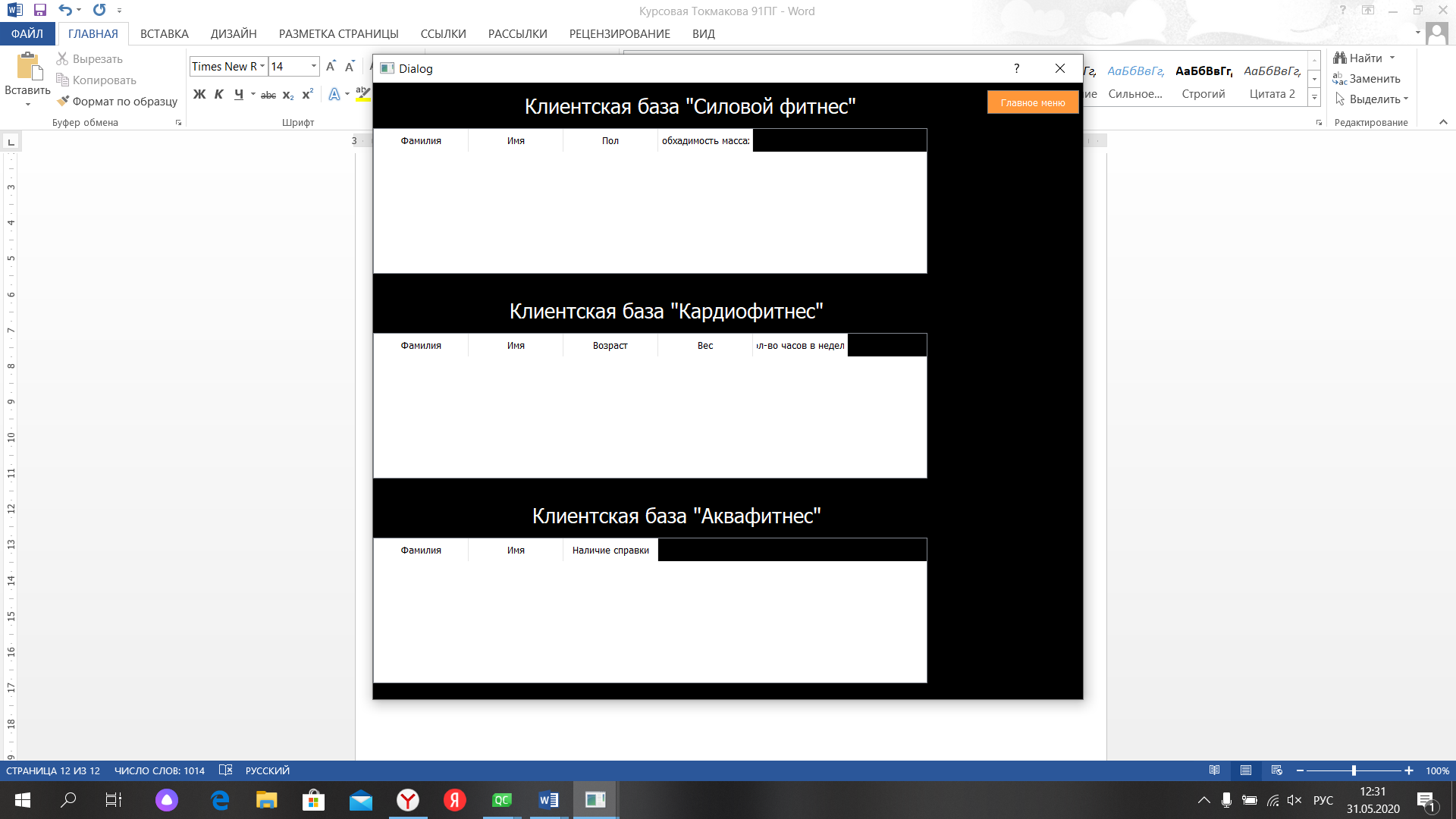


Рисунок 17 – Окно кнопки «Поиск по»

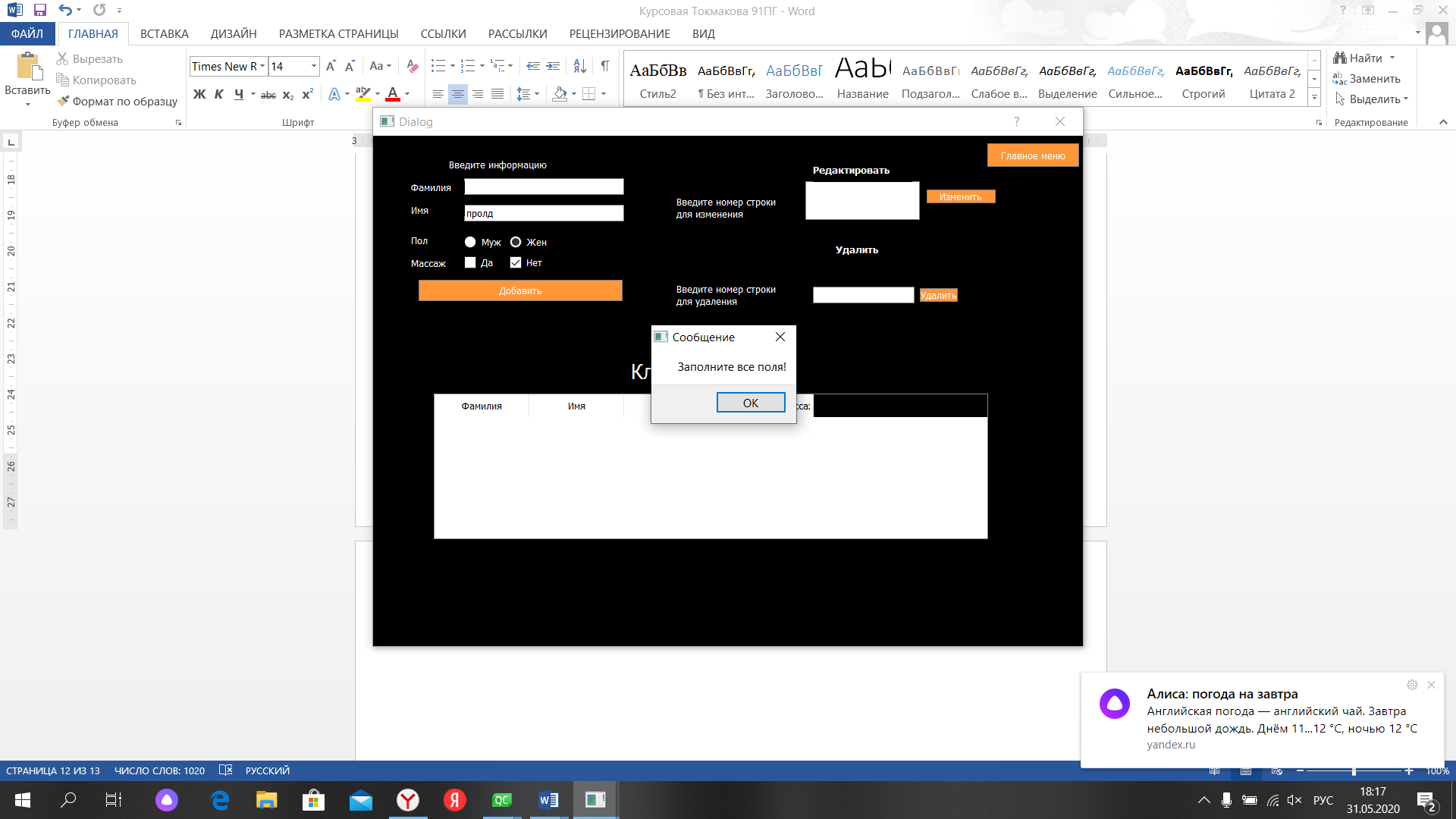


Рисунок 18 – Сообщение об ошибке при незаполненных полях

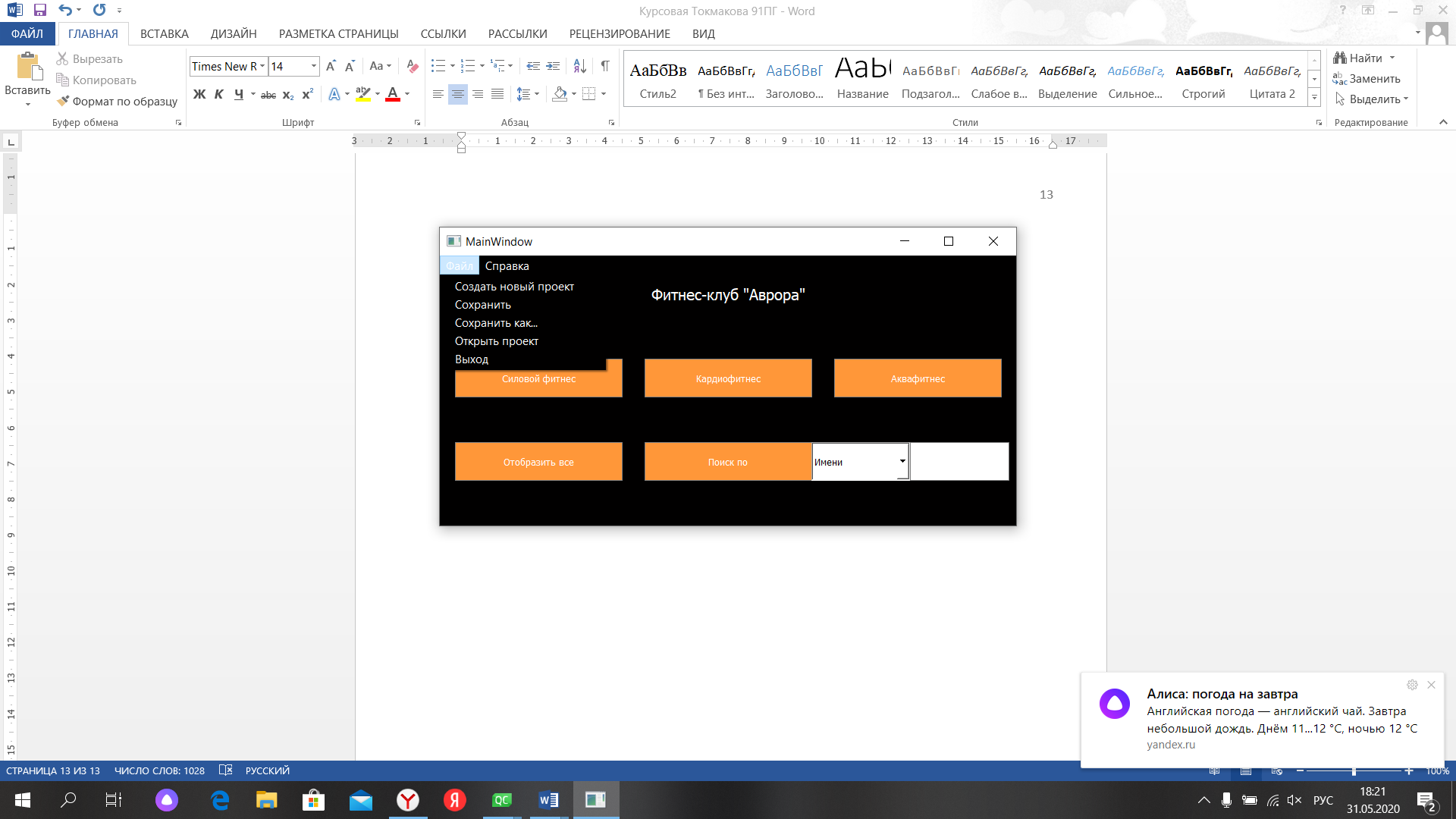


Рисунок 19 – Окно кнопки «Файл»

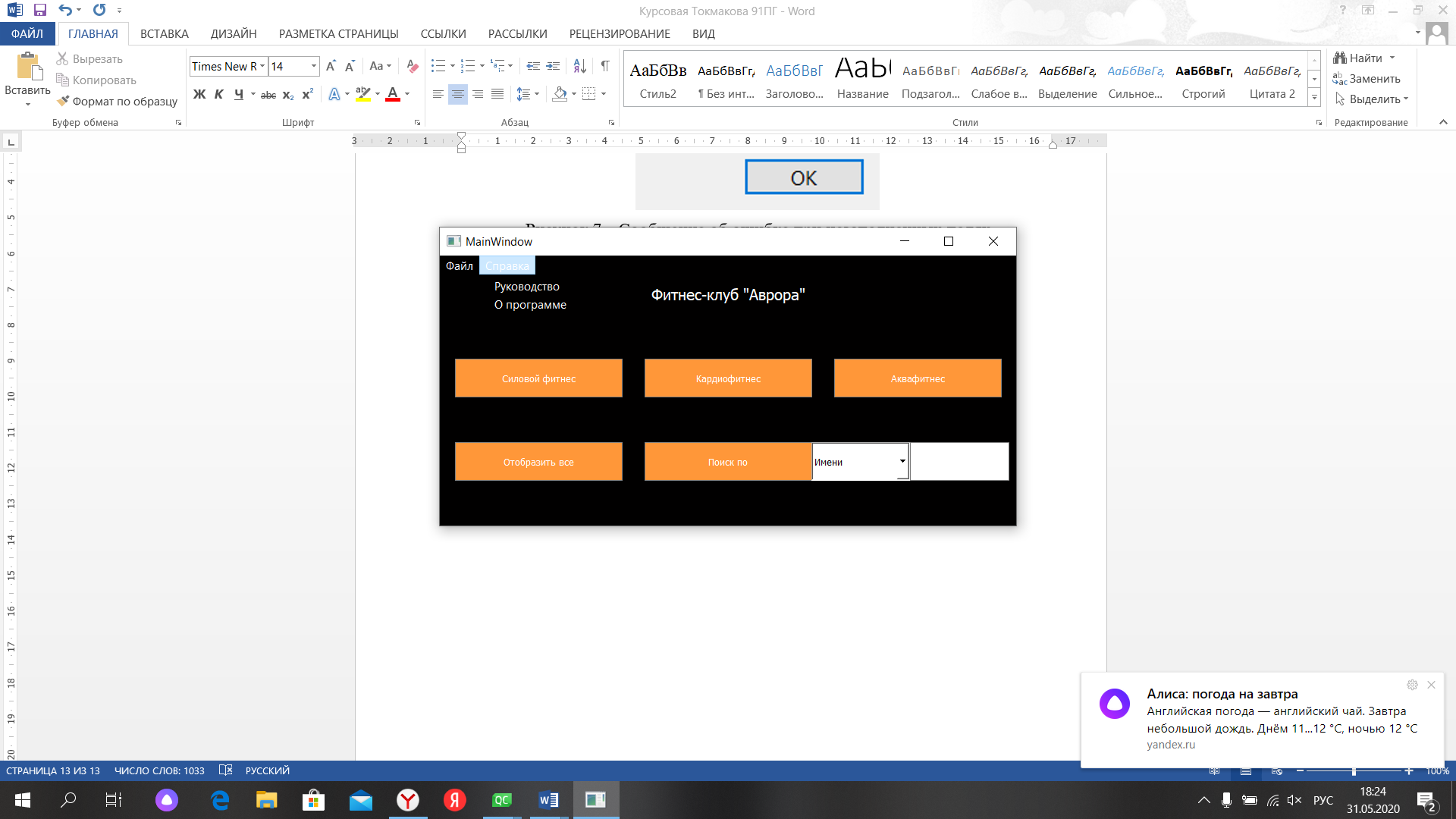


Рисунок 20 – Окно кнопки «Справка»

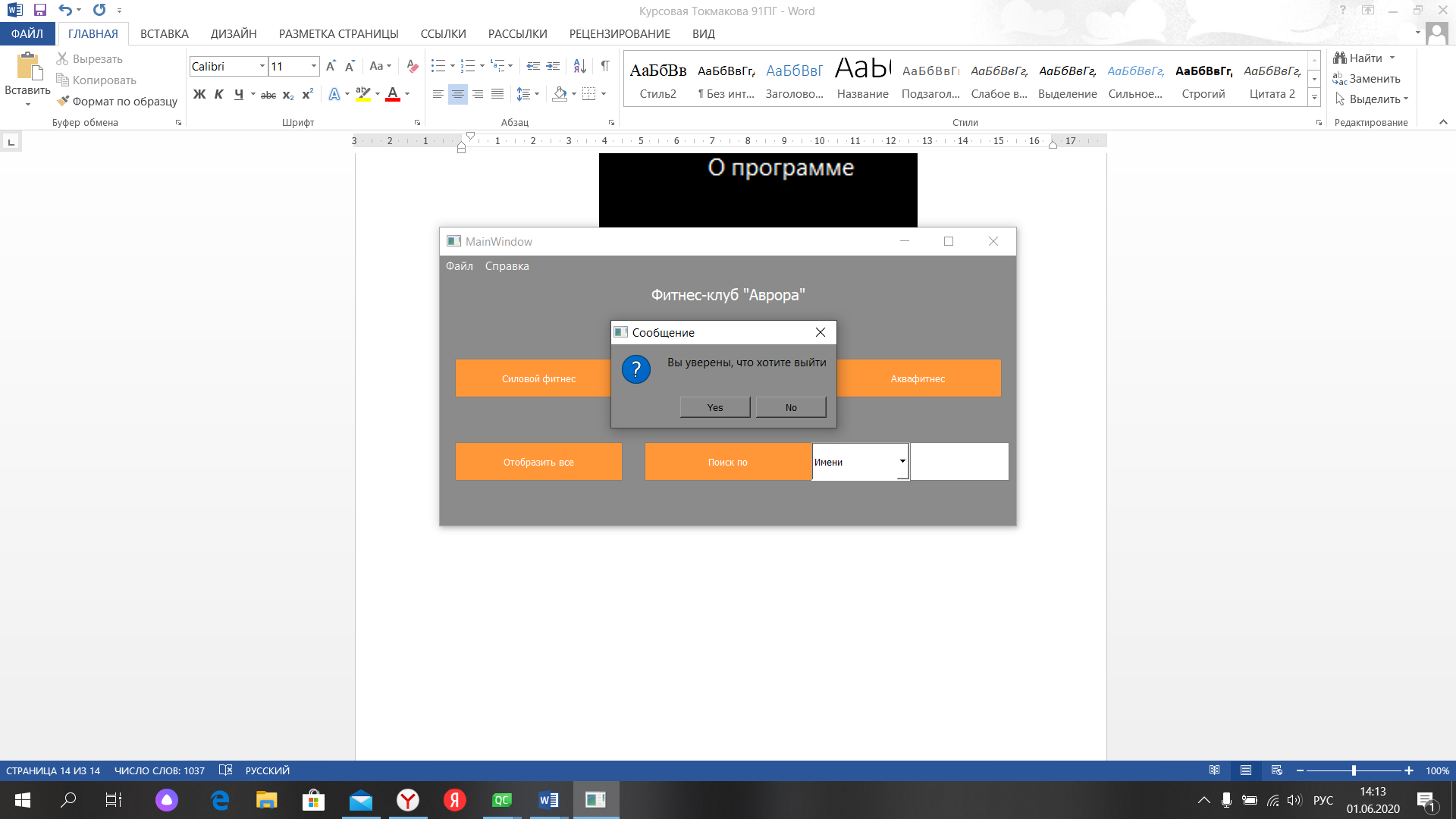


Рисунок 21 – Окно выхода

Кнопка «Выход» организована в виде дружественного выхода, как показано на рисунке 21.

Для того, чтобы открыть сохраненный проект, пользователю нужно выбрать пункт меню «Открыть проект», после чего появится диалоговое окно для выбора файла. Диалоговое окно для выбора файла показано на рисунке 22.

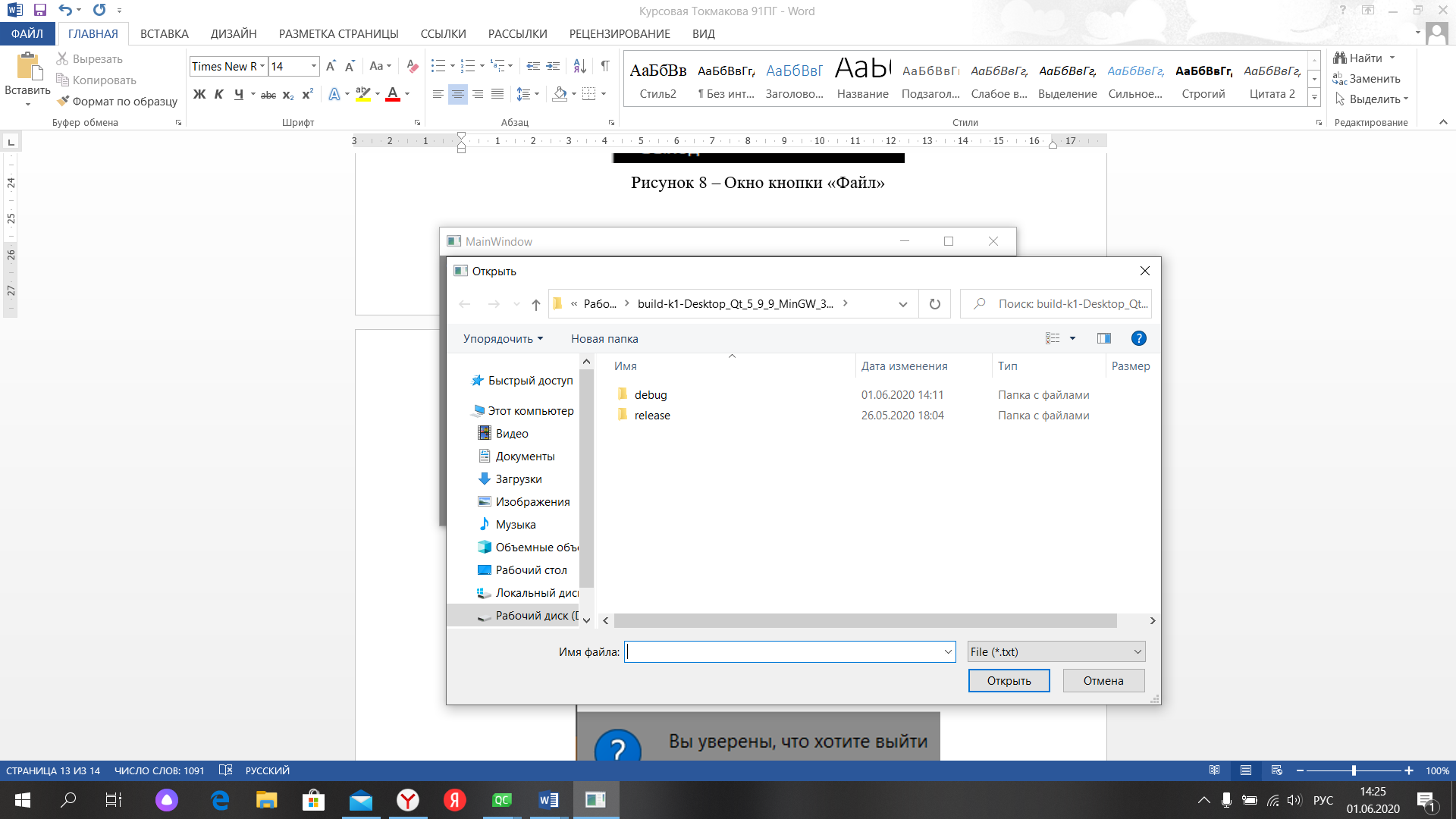


Рисунок 22 – Окно для открытия файла

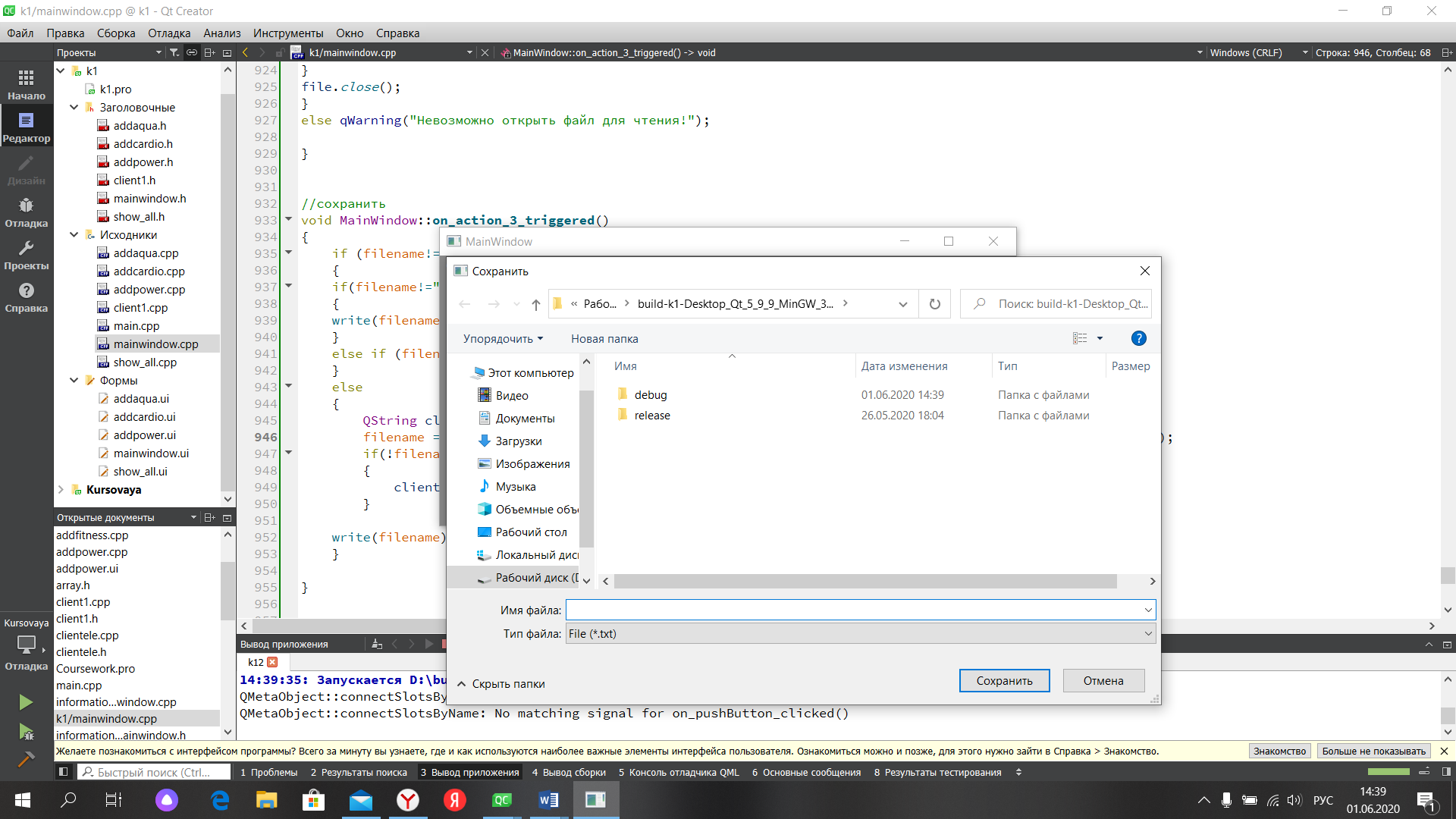


Рисунок 23 – Окно для сохранения файла

# ЗАКЛЮЧЕНИЕ

В ходе выполнения проекта мы познакомились с различными структурами данных, выявили их достоинства и недостатки; разработали алгоритмы для решения поставленных задач и реализовали их на языке высокого уровня С++, подробно описали особенности программной реализации и пользовательский интерфейс.

Поскольку метод решения и структуры данных выбраны, алгоритмы решения поставленных задач разработаны, пользовательский интерфейс описан, программа на языке С++ с использованием объектно-ориентированного программирования реализована, то поставленную цель можно считать выполненной.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Ашарина, И.В. Объектно-ориентированное программирование в С++: лекции и упражнения: Учебное пособие для вузов / И.В. Ашарина. - М.: РиС, 2015. - 336 c.
2. Васильев, А.Н. Объектно-ориентированное программирование на C++ / А.Н. Васильев. - СПб.: Наука и техника, 2016. - 544 c.
3. Динамические массивы и переменные: легко и просто. [Электронный ресурс]. – Режим доступа: [https://codelessons.ru/cplusplus/dinamicheskie-massiv..](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fcodelessons.ru%2Fcplusplus%2Fdinamicheskie-massivy-i-peremennye-vse-samoe-glavnoe.html) (дата обращения: 18.05.2020)
4. Лафоре, Р. Объектно-ориентированное программирование в С++ / Р. Лафоре. - СПб.: Питер, 2019. - 928 c.
5. Лучшие статьи для программистов: C#, Python, паттерны проектирования, алгоритмы и структуры данных [Электронный ресурс]. – Режим доступа: [https://shwanoff.ru/dynamic-structures-cpp/](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fshwanoff.ru%2Fdynamic-structures-cpp%2F) (дата обращения 01.05.20).
6. Павловская, Т. С/С++. Процедурное и объектно-ориентированное программирование / Т. Павловская. - СПб.: Питер, 2018. - 496 c.
7. Радченко, Г.И. Объектно-ориентированное программирование  
   [Текст] / Г.И. Радченко, Е.А. Захаров. - Челябинск: Издательский центр ЮУрГУ, 2013. –167 с.
8. Свободная энциклопедия Википедия. Структура данных. [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Структура\_данных](https://vk.com/away.php?utf=1&to=https%3A%2F%2Fru.wikipedia.org%2Fwiki%2F%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) (дата обращения: 18.05.2020)
9. Сообщество IT специалистов Хабр. О выборе структур данных для начинающих [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/339656/> (дата обращения 18.05.20).

# ПРИЛОЖЕНИЕ А

# (обязательное)

# ЛИСТИНГ ФАЙЛА “mainwindow.cpp”

#include "mainwindow.h"

#include "ui\_mainwindow.h"

#include "client1.h"

#include "addaqua.h"

#include "addcardio.h"

#include "addpower.h"

#include <QDebug>

#include <iostream>

#include "show\_all.h"

#include <QMessageBox>

#include <QFile>

#include <QFileDialog>

using namespace std;

MainWindow::**MainWindow**(QWidget \*parent)

: QMainWindow(parent)

, ui(new Ui::MainWindow)

{

ui->setupUi(this);

}

Cl \*\*cli=new Cl \*[0];

int n=0,power1=0,aqua1=0,cardio1=0;

int o=0;

QString filename; QString filename1;

QString surname;

MainWindow::~***MainWindow***()

{

delete ui;

}

void MainWindow::**on\_pushButton\_power\_clicked**()

{

addpower g;

if (power1>0)

{

for (int i=0;i<n;i++)

{

QString A;

((power\*)cli[i])->gettype(*A*);

if(A=="Силовой фитнес")

{

QString B,C,D;

((power\*)cli[i])->setInfo(*A*,*B*);

((power\*)cli[i])->set\_power(*C*,*D*);

g.set(*A*,*B*,*C*,*D*);

}

}

}

QString A,B,C,D;

g.*exec*();

int k;

int count2=0;

g.set\_num(*k*);

if (power1>0)

{

int m=power1-k;

if (m==0)

{

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Силовой фитнес")

{

g.tpower(*A*,*B*,*C*,*D*,count2);

count2++;

QString E="Силовой фитнес";

cli[i]=new power(A,B,E,C,D);

}

}

}

else if(m>0)

{

if (k==0)

{

int r1=power1;

for(int z=0;z<r1;z++)

{

o=0;

for(int i=0;i<n-1;i++)

{

QString N2;

cli[i]->gettype(*N2*);

if (N2=="Силовой фитнес")

{

if(o==power1-1)

{

for (int j=i;j<n-1;j++)

{

cli[j]=cli[j+1];

o+=3;

}

}

else o++;

}

}

n--;

power1--;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

}

}

else

{

int r=m;

for(int z=0;z<r;z++)

{

o=0;

for(int i=0;i<n-1;i++)

{

QString N2;

cli[i]->gettype(*N2*);

if (N2=="Силовой фитнес")

{

if(o==m)

{

for (int j=i;j<n-1;j++)

{

cli[j]=cli[j+1];

o+=3;

}

}

else o++;

}

}

m--;

n--;

power1--;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

}

}

int count=0;

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Силовой фитнес")

{

g.tpower(*A*,*B*,*C*,*D*,count);

count++;

QString E="Силовой фитнес";

cli[i]=new power(A,B,E,C,D);

if (count==m) break;

}

}

}

else

{

int count=m\*(-1);

n=n+count;

power1=power1+count;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

for (int i=n-count;i<n;i++)

{

QString E="Силовой фитнес";

cli[i]=new power(A,B,E,C,D);

}

count=0;

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Силовой фитнес")

{

g.tpower(*A*,*B*,*C*,*D*,count);

count++;

QString E="Силовой фитнес";

cli[i]=new power(A,B,E,C,D);

}

}

}

}

else

{

int count=0;

for (int i=power1;i<k;i++)

{

g.tpower(*A*,*B*,*C*,*D*,count);

count++;

n++;

power1++;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

QString E="Силовой фитнес";

cli[n-1]=new power(A,B,E,C,D);

}

}

g.zero();

}

void MainWindow::**on\_pushButton\_kardio\_clicked**()

{

addcardio g;

if (cardio1>0)

{

for (int i=0;i<n;i++)

{

QString A;

((cardio\*)cli[i])->gettype(*A*);

if(A=="Кардио")

{

QString B,C,D,E;

((cardio\*)cli[i])->setInfo(*A*,*B*);

((cardio\*)cli[i])->set\_cardio(*C*,*D*,*E*);

g.set2(*A*,*B*,*C*,*D*,*E*);

}

}

}

QString A,B,C,D,E;

g.*exec*();

int k=0;

int count=0;

g.set\_num2(*k*);

if (cardio1>0)

{

int m=cardio1-k;

if (m==0)

{

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Кардио")

{

g.tcardio(*A*,*B*,*C*,*D*,*E*,count);

count++;

QString F="Кардио";

cli[i]=new cardio(A,B,F,C,D,E);

}

}

}

else if(m>0)

{

if (k==0)

{

int r1=cardio1;

for(int z=0;z<r1;z++)

{

o=0;

for(int i=0;i<n-1;i++)

{

QString N2;

cli[i]->gettype(*N2*);

if (N2=="Кардио")

{

if(o==cardio1-1)

{

for (int j=i;j<n-1;j++)

{

cli[j]=cli[j+1];

o+=3;

}

}

else o++;

}

}

n--;

cardio1--;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

}

}

else

{

int r=m;

for(int z=0;z<r;z++)

{

o=0;

for(int i=0;i<n-1;i++)

{

QString N2;

cli[i]->gettype(*N2*);

if (N2=="Кардио")

{

if(o==m)

{

for (int j=i;j<n-1;j++)

{

cli[j]=cli[j+1];

o+=3;

}

}

else o++;

}

}

m--;

n--;

cardio1--;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

}

}

int count=0;

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Кардио")

{

g.tcardio(*A*,*B*,*C*,*D*,*E*,count);

count++;

QString F="Кардио";

cli[i]=new cardio(A,B,F,C,D,E);

if (count==m) break;

}

}

}

else

{

int count=m\*(-1);

n=n+count;

cardio1=cardio1+count;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

for (int i=n-count;i<n;i++)

{

QString F="Кардио";

cli[i]=new cardio(A,B,F,C,D,E);

}

count=0;

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Кардио")

{

g.tcardio(*A*,*B*,*C*,*D*,*E*,count);

count++;

QString F="Кардио";

cli[i]=new cardio(A,B,F,C,D,E);

}

}

}

}

else

{

count=0;

for (int i=cardio1;i<k;i++)

{

g.tcardio(*A*,*B*,*C*,*D*,*E*,count);

count++;

n++;

cardio1++;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

QString F="Кардио";

cli[n-1]=new cardio(A,B,F,C,D,E);

}

}

g.zero2();

}

void MainWindow::**on\_pushButton\_aqua\_clicked**()

{

addaqua g;

if (aqua1>0)

{

for (int i=0;i<n;i++)

{

QString A;

((aqua\*)cli[i])->gettype(*A*);

if(A=="Аквафитнес")

{

QString B,C;

((aqua\*)cli[i])->setInfo(*A*,*B*);

((aqua\*)cli[i])->set\_aqua(*C*);

g.set3(*A*,*B*,*C*);

}

}

}

QString A,B,C;

g.*exec*();

int k=0;

int count=0;

g.set\_num3(*k*);

if (aqua1>0)

{

int m=aqua1-k;

if (m==0)

{

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Аквафитнес")

{

g.taqua(*A*,*B*,*C*,count);

count++;

QString F="Аквафитнес";

cli[i]=new aqua(A,B,F,C);

}

}

}

else if(m>0)

{

if (k==0)

{

int r1=aqua1;

for(int z=0;z<r1;z++)

{

o=0;

for(int i=0;i<n-1;i++)

{

QString N2;

cli[i]->gettype(*N2*);

if (N2=="Аквафитнес")

{

if(o==aqua1-1)

{

for (int j=i;j<n-1;j++)

{

cli[j]=cli[j+1];

o+=3;

}

}

else o++;

}

}

n--;

aqua1--;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

}

}

else

{

int r=m;

for(int z=0;z<r;z++)

{

o=0;

for(int i=0;i<n-1;i++)

{

QString N2;

cli[i]->gettype(*N2*);

if (N2=="Аквафитнес")

{

if(o==m)

{

for (int j=i;j<n-1;j++)

{

cli[j]=cli[j+1];

o+=3;

}

}

else o++;

}

}

m--;

n--;

aqua1--;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

}

}

int count=0;

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Аквафитнес")

{

g.taqua(*A*,*B*,*C*,count);

count++;

QString F="Аквафитнес";

cli[i]=new aqua(A,B,F,C);

if (count==m) break;

}

}

}

else

{

int count=m\*(-1);

n=n+count;

aqua1=aqua1+count;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

for (int i=n-count;i<n;i++)

{

QString F="Аквафитнес";

cli[i]=new aqua(A,B,F,C);

}

count=0;

for (int i=0;i<n;i++)

{

QString N;

cli[i]->gettype(*N*);

if (N=="Аквафитнес")

{

g.taqua(*A*,*B*,*C*,count);

count++;

QString F="Аквафитнес";

cli[i]=new aqua(A,B,F,C);

}

}

}

}

else

{

count=0;

for (int i=aqua1;i<k;i++)

{

g.taqua(*A*,*B*,*C*,count);

count++;

n++;

aqua1++;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

QString F="Аквафитнес";

cli[n-1]=new aqua(A,B,F,C);

}

}

g.zero3();

}

void MainWindow::**on\_pushButton\_power\_2\_clicked**()

{

show\_all g;

QString type;

for (int i=0;i<n;i++)

{

cli[i]->gettype(*type*);

if(type=="Силовой фитнес")

{

QString A,B,C,D;

cli[i]->setInfo(*A*,*B*);

((power\*)cli[i])->set\_power(*C*,*D*);

g.get(*A*,*B*,*C*,*D*);

}

else if (type=="Кардио")

{

QString A,B,C,D,E;

cli[i]->setInfo(*A*,*B*);

((cardio\*)cli[i])->set\_cardio(*C*,*D*,*E*);

g.get2(*A*,*B*,*C*,*D*,*E*);

}

else

{

QString A,B,C;

cli[i]->setInfo(*A*,*B*);

((aqua\*)cli[i])->set\_aqua(*C*);

g.get3(*A*,*B*,*C*);

}

}

g.*exec*();

g.zero();

}

void MainWindow::**on\_pushButton\_power\_3\_clicked**()

{

show\_all g;

if (ui->lineEdit->text()=="")

{

}

else{

QString A=ui->comboBox->currentText();

QString D=ui->lineEdit->text();

if (A=="Фамилии")

{

for (int i=0;i<n;i++)

{

QString name;

QString type;

cli[i]->gettype(*type*);

if(type=="Силовой фитнес")

{

cli[i]->getname(*name*);

if (name==D)

{

QString A,B,C,D;

cli[i]->setInfo(*A*,*B*);

((power\*)cli[i])->set\_power(*C*,*D*);

g.get(*A*,*B*,*C*,*D*);

}

}

}

for (int i=0;i<n;i++)

{

QString name;

QString type;

cli[i]->gettype(*type*);

if(type=="Кардио")

{

cli[i]->getname(*name*);

if (name==D)

{

QString A,B,C,D,E;

cli[i]->setInfo(*A*,*B*);

((cardio\*)cli[i])->set\_cardio(*C*,*D*,*E*);

g.get2(*A*,*B*,*C*,*D*,*E*);

}

}

}

for (int i=0;i<n;i++)

{

QString name;

QString type;

cli[i]->gettype(*type*);

if(type=="Аквафитнес")

{

cli[i]->getname(*name*);

if (name==D)

{

QString A,B,C;

cli[i]->setInfo(*A*,*B*);

((aqua\*)cli[i])->set\_aqua(*C*);

g.get3(*A*,*B*,*C*);

}

}

}

}

else if (A=="Имени")

{

for (int i=0;i<n;i++)

{

QString type;

cli[i]->gettype(*type*);

if(type=="Силовой фитнес")

{

cli[i]->getsurname(*surname*);

if (surname==D)

{

QString A,B,C,D;

cli[i]->setInfo(*A*,*B*);

((power\*)cli[i])->set\_power(*C*,*D*);

g.get(*A*,*B*,*C*,*D*);

}

}

}

for (int i=0;i<n;i++)

{

QString type;

cli[i]->gettype(*type*);

if(type=="Кардио")

{

cli[i]->getsurname(*surname*);

if (surname==D)

{

QString A,B,C,D,E;

cli[i]->setInfo(*A*,*B*);

((cardio\*)cli[i])->set\_cardio(*C*,*D*,*E*);

g.get2(*A*,*B*,*C*,*D*,*E*);

}

}

}

for (int i=0;i<n;i++)

{

QString type;

cli[i]->gettype(*type*);

if(type=="Аквафитнес")

{

cli[i]->getsurname(*surname*);

if (surname==D)

{

QString A,B,C;

cli[i]->setInfo(*A*,*B*);

((aqua\*)cli[i])->set\_aqua(*C*);

g.get3(*A*,*B*,*C*);

}

}

}

}

else

{

int num=D.toUInt()-1;

int count=0;

for (int j=0;j<n;j++)

{

QString type;

cli[j]->gettype(*type*);

if(type=="Силовой фитнес")

{

if (count==num)

{

QString A,B,C,D;

cli[j]->setInfo(*A*,*B*);

((power\*)cli[j])->set\_power(*C*,*D*);

g.get(*A*,*B*,*C*,*D*);

count+=100;

}

else count++;

}

}

count=0;

for (int j=0;j<n;j++)

{

QString type;

cli[j]->gettype(*type*);

if(type=="Кардио")

{

if (count==num)

{

QString A,B,C,D,E;

cli[j]->setInfo(*A*,*B*);

((cardio\*)cli[j])->set\_cardio(*C*,*D*,*E*);

g.get2(*A*,*B*,*C*,*D*,*E*);

count+=100;

}

else count++;

}

}

count=0;

for (int j=0;j<n;j++)

{

QString type;

cli[j]->gettype(*type*);

if(type=="Аквафитнес")

{

if (count==num)

{

QString A,B,C;

cli[j]->setInfo(*A*,*B*);

((aqua\*)cli[j])->set\_aqua(*C*);

g.get3(*A*,*B*,*C*);

count+=100;

}

else count++;

}

}

}

}

g.*exec*();

g.zero();

}

void MainWindow::**on\_action\_triggered**()

{

QMessageBox::StandardButton reply;

reply = QMessageBox::question(this, "Сообщение", "Вы уверены, что хотите выйти",

QMessageBox::Yes|QMessageBox::No);

if (reply == QMessageBox::Yes) {

QApplication::quit();

} else {

}

}

void MainWindow::**on\_action\_7\_triggered**()

{

QMessageBox::information(this,"Руководство","При редактировании информации выбранная строка, если она существует, выделяется зеленым цветом,для изменения информации нажмите кнопку Добавить."

"\nПри редактировании и удалении, если ввести число превышающее количество строк в таблице, ничего не произойдет."

"\nФайлы сохраняются в формате txt"

);

}

void MainWindow::**on\_action\_8\_triggered**()

{

QMessageBox::information(this,"О программе","Выполнила студентка группы 91ПГ Токмакова Наталья");

}

void MainWindow::**on\_action\_2\_triggered**()

{

delete [] cli;

n=0,power1=0,aqua1=0,cardio1=0;

o=0;

surname="";

filename="";

filename1="";

}

void MainWindow::**write** (QString filename)

{

QFile file(filename);

if (file.*open*(QIODevice::WriteOnly)) {

QTextStream out(&file);

for (int i=0;i<n;i++)

{

QString type;

cli[i]->gettype(*type*);

if(type=="Аквафитнес")

{

QString A,B,C;

cli[i]->setInfo(*A*,*B*);

((aqua\*)cli[i])->set\_aqua(*C*);

out << type << endl;

out << A << endl;

out << B << endl;

out << C << endl;

}

else if(type=="Кардио")

{

QString A,B,C,D,E;

cli[i]->setInfo(*A*,*B*);

((cardio\*)cli[i])->set\_cardio(*C*,*D*,*E*);

out << type << endl;

out << A << endl;

out << B << endl;

out << C << endl;

out << D << endl;

out << E << endl;

}

else

{

QString A,B,C,D;

cli[i]->setInfo(*A*,*B*);

((power\*)cli[i])->set\_power(*C*,*D*);

out << type << endl;

out << A << endl;

out << B << endl;

out << C << endl;

out << D << endl;

}

}

}

else qWarning("Не удалось открыть файл");

file.*close*();

}

void MainWindow:: **read** (QString filename)

{

QFile file(filename);

if(file.*open*(QIODevice::ReadOnly |QIODevice::Text)){

QTextStream in(&file);

while (!in.atEnd())

{

QString type = in.readLine();

if (type=="Аквафитнес")

{

QString A,B,C;

A=in.readLine();

B=in.readLine();

C=in.readLine();

n++;

aqua1++;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

cli[n-1]=new aqua(A,B,type,C);

}

else if (type=="Кардио")

{

QString A,B,C,D,E;

A=in.readLine();

B=in.readLine();

C=in.readLine();

D=in.readLine();

E=in.readLine();

n++;

cardio1++;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

cli[n-1]=new cardio(A,B,type,C,D,E);

}

else if (type=="Силовой фитнес")

{

QString A,B,C,D;

A=in.readLine();

B=in.readLine();

C=in.readLine();

D=in.readLine();

n++;

power1++;

Cl \*\*P = new Cl\*[n];

for (int i = 0; i<n; i++)

{ P[i] = cli[i]; }

delete[] cli;

cli = P;

cli[n-1]=new power(A,B,type,C,D);

}

}

file.*close*();

}

else qWarning("Невозможно открыть файл для чтения!");

}

//сохранить

void MainWindow::**on\_action\_3\_triggered**()

{

if (filename!="" || filename1!="")

{

if(filename!="")

{

write(filename);

}

else if (filename1!="") write(filename1);

}

else

{

QString clientfile;

filename = QFileDialog::getSaveFileName(this, tr("Сохранить"), QDir::currentPath(), tr("File (\*.txt)") );

if(!filename.isEmpty())

{

clientfile=filename;

}

write(filename);

}

}

//сохранить как

void MainWindow::**on\_action\_4\_triggered**()

{

QString clientfile;

filename = QFileDialog::getSaveFileName(this, tr("Сохранить как"), QDir::currentPath(), tr("File (\*.txt)") );

if(!filename.isEmpty())

{

clientfile=filename;

}

write (filename);

}

//открыть

void MainWindow::**on\_action\_5\_triggered**()

{

delete [] cli;

n=0,power1=0,aqua1=0,cardio1=0;

o=0;

surname="";

filename="";

filename1="";

QString clientfile;

filename1 = QFileDialog::getOpenFileName(this, tr("Открыть"), QDir::currentPath(), tr("File (\*.txt)"));

if(!filename1.isEmpty())

{

clientfile=filename1;

}

read (filename1);

}

# ПРИЛОЖЕНИЕ B

# (обязательное)

# ЛИСТИНГ ФАЙЛА “mainwindow.h”

#ifndef MAINWINDOW\_H

#define MAINWINDOW\_H

#include <QMainWindow>

QT\_BEGIN\_NAMESPACE

namespace **Ui** { class **MainWindow**; }

QT\_END\_NAMESPACE

class **MainWindow** : public QMainWindow

{

Q\_OBJECT

public:

**MainWindow**(QWidget \*parent = nullptr);

~***MainWindow***();

void **write** (QString filename);

void **read** (QString filename);

private slots:

void **on\_pushButton\_2\_clicked**();

void **on\_pushButton\_clicked**();

void **on\_pushButton\_power\_clicked**();

void **on\_pushButton\_kardio\_clicked**();

void **on\_pushButton\_aqua\_clicked**();

void **on\_pushButton\_power\_2\_clicked**();

void **on\_pushButton\_power\_3\_clicked**();

void **on\_action\_triggered**();

void **on\_action\_7\_triggered**();

void **on\_action\_8\_triggered**();

void **on\_action\_2\_triggered**();

void **on\_action\_3\_triggered**();

void **on\_action\_4\_triggered**();

void **on\_action\_5\_triggered**();

private:

Ui::MainWindow \*ui;

};

#endif // MAINWINDOW\_H

# ПРИЛОЖЕНИЕ C

# (обязательное)

# ЛИСТИНГ ФАЙЛА «addaqua.cpp»

#include "addaqua.h"

#include "ui\_addaqua.h"

#include <QMessageBox>

addaqua::**addaqua**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::addaqua)

{

ui->setupUi(this);

QRegExp Const\_v ("^[А-Яа-я]{,50}$");

QRegExpValidator \*const\_all = new QRegExpValidator(Const\_v, this);

ui->lineEdit\_surname->setValidator(const\_all);

ui->lineEdit\_name->setValidator(const\_all);

QRegExp Const\_dd ("^[0-9]+$");

QRegExpValidator \*const\_delete = new QRegExpValidator(Const\_dd, this);

ui->lineEdit\_delete->setValidator(const\_delete);

ui->lineEdit\_edit->setValidator(const\_delete);

}

addaqua::~***addaqua***()

{

delete ui;

}

int a3=0,b3=0;

int selection2=-1;

void addaqua::**on\_pushButton\_clicked**()

{

if(selection2>-1)

{

if(ui->lineEdit\_surname->text()==""||ui->lineEdit\_name->text()=="" || (!ui->checkBox->isChecked() && !ui->checkBox\_2->isChecked())||(ui->checkBox->isChecked() && ui->checkBox\_2->isChecked()))

{

QMessageBox msgBox;

msgBox.setWindowTitle("Сообщение");

msgBox.setText("Заполните все поля!");

msgBox.*exec*();

}

else

{

QString A,B,C;

A=ui->lineEdit\_surname->text();

B=ui->lineEdit\_name->text();

if(ui->checkBox->isChecked()){

C=ui->checkBox->text();

}

if(ui->checkBox\_2->isChecked()){

C=ui->checkBox\_2->text();

}

ui->tableWidget->setItem(selection2,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(selection2,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(selection2,2,new QTableWidgetItem(C));

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,0)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,1)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,2)->setBackground(Qt::white);

}

}

else

{

if(ui->lineEdit\_surname->text()==""||ui->lineEdit\_name->text()=="" || (!ui->checkBox->isChecked() && !ui->checkBox\_2->isChecked())||(ui->checkBox->isChecked() && ui->checkBox\_2->isChecked()))

{

QMessageBox msgBox;

msgBox.setWindowTitle("Сообщение");

msgBox.setText("Заполните все поля!");

msgBox.*exec*();

}

else

{

int count=0;

QString A,B,C;

A=ui->lineEdit\_surname->text();

B=ui->lineEdit\_name->text();

if(ui->checkBox->isChecked()){

C=ui->checkBox->text();

}

if(ui->checkBox\_2->isChecked()){

C=ui->checkBox\_2->text();

}

for (int i=0;i<a3;i++)

{

QString A2,B2,C2;

A2=ui->tableWidget->item(i,0)->text();

B2=ui->tableWidget->item(i,1)->text();

C2=ui->tableWidget->item(i,2)->text();

if (A==A2 && B==B2 && C==C2)

{

count++;

}

}

if (count>0)

{

QMessageBox::warning(this,"Ошибка","такой эемент уже есть!");

count=0;

}

else

{

a3++;

ui->tableWidget->setRowCount(a3);

ui->tableWidget->setItem(b3,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(b3,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(b3,2,new QTableWidgetItem(C));

b3++;

}

}

}

selection2=-1;

}

void addaqua::**taqua**(QString &A, QString &B, QString &C, int l3)

{

A=ui->tableWidget->item(l3,0)->text();

B=ui->tableWidget->item(l3,1)->text();

C=ui->tableWidget->item(l3,2)->text();

}

void addaqua::**set\_num3**(int &num)

{

num=a3;

}

void addaqua::**set3**(QString &A, QString &B, QString &C)

{

a3++;

ui->tableWidget->setRowCount(a3);

ui->tableWidget->setItem(b3,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(b3,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(b3,2,new QTableWidgetItem(C));

b3++;

}

void addaqua::**zero3**()

{

a3=0;b3=0;

}

void addaqua::**on\_pushButton\_edit\_clicked**()

{

if (ui->lineEdit\_edit->text().toUInt()<=a3)

{

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,0)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,1)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,2)->setBackground(Qt::green);

selection2=ui->lineEdit\_edit->text().toInt()-1;

}

}

void addaqua::**on\_pushButton\_edit\_2\_clicked**()

{

this->close();

}

void addaqua::**on\_pushButton\_delete\_clicked**()

{

for(int i=0;i<=a3-1;i++)

{

if (i==ui->lineEdit\_delete->text().toInt()-1)

{

for (int j=i;j<a3-1;j++)

{

QString A,B,C;

A=ui->tableWidget->item(j+1,0)->text();

B=ui->tableWidget->item(j+1,1)->text();

C=ui->tableWidget->item(j+1,2)->text();

ui->tableWidget->setItem(j,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(j,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(j,2,new QTableWidgetItem(C));

}

a3--;

b3--;

ui->tableWidget->setRowCount(a3);

}

}

}

# ПРИЛОЖЕНИЕ D

# (обязательное)

# ЛИСТИНГ ФАЙЛА “addaqua.h”

#ifndef ADDAQUA\_H

#define ADDAQUA\_H

#include <QDialog>

namespace **Ui** {

class **addaqua**;

}

class **addaqua** : public QDialog

{

Q\_OBJECT

public:

explicit **addaqua**(QWidget \*parent = nullptr);

~***addaqua***();

void **taqua**(QString &A,QString &B,QString &C, int l3);

void **set\_num3** (int &num);

void **set3**(QString &A,QString &B,QString &C);

void **zero3**();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_edit\_clicked**();

void **on\_pushButton\_edit\_2\_clicked**();

void **on\_pushButton\_delete\_clicked**();

private:

Ui::addaqua \*ui;

};

#endif // ADDAQUA\_H

# ПРИЛОЖЕНИЕ E

# (обязательное)

# ЛИСТИНГ ФАЙЛА “addcardio.cpp”

#include "addcardio.h"

#include "ui\_addcardio.h"

#include <QMessageBox>

addcardio::**addcardio**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::addcardio)

{

ui->setupUi(this);

QRegExp ipRegex ("^[1-9]{1}[0-9]{1}$");

QRegExpValidator \*ipValidator = new QRegExpValidator(ipRegex, this);

ui->lineEdit\_age->setValidator(ipValidator);

ui->lineEdit\_weight->setValidator(ipValidator);

QRegExp Const\_v ("^[А-Яа-я]{,50}$");

QRegExpValidator \*const\_all = new QRegExpValidator(Const\_v, this);

ui->lineEdit\_surname->setValidator(const\_all);

ui->lineEdit\_name->setValidator(const\_all);

QRegExp Const\_dd ("^[0-9]+$");

QRegExpValidator \*const\_delete = new QRegExpValidator(Const\_dd, this);

ui->lineEdit\_delete->setValidator(const\_delete);

ui->lineEdit\_edit->setValidator(const\_delete);

}

addcardio::~***addcardio***()

{

delete ui;

}

int a2=0,b2=0;

int selection1=-1;

int count\_del1=0;

void addcardio::**tcardio**(QString &A, QString &B, QString &C, QString &D, QString &E,int l2)

{

A=ui->tableWidget->item(l2,0)->text();

B=ui->tableWidget->item(l2,1)->text();

C=ui->tableWidget->item(l2,2)->text();

D=ui->tableWidget->item(l2,3)->text();

E=ui->tableWidget->item(l2,4)->text();

}

void addcardio::**set\_num2**(int &num)

{

num=a2;

}

void addcardio::**set2**(QString &A, QString &B, QString &C, QString &D, QString &E)

{

a2++;

ui->tableWidget->setRowCount(a2);

ui->tableWidget->setItem(b2,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(b2,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(b2,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(b2,3,new QTableWidgetItem(D));

ui->tableWidget->setItem(b2,4,new QTableWidgetItem(E));

b2++;

}

void addcardio::**zero2**()

{

a2=0;b2=0;count\_del1=0;

}

void addcardio::**on\_pushButton\_clicked**()

{

if(selection1>-1)

{

if(ui->lineEdit\_surname->text()==""||ui->lineEdit\_name->text()==""||ui->lineEdit\_age->text()==""||ui->lineEdit\_weight->text()=="")

{

QMessageBox msgBox;

msgBox.setWindowTitle("Сообщение");

msgBox.setText("Заполните все поля!");

msgBox.*exec*();

}

else

{

QString A,B,C,D,E;

A=ui->lineEdit\_surname->text();

B=ui->lineEdit\_name->text();

C=ui->lineEdit\_age->text();

D=ui->lineEdit\_weight->text();

E=ui->comboBox->currentText();

ui->tableWidget->setItem(selection1,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(selection1,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(selection1,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(selection1,3,new QTableWidgetItem(D));

ui->tableWidget->setItem(selection1,4,new QTableWidgetItem(E));

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,0)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,1)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,2)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,3)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,4)->setBackground(Qt::white);

}

}

else

{

if(ui->lineEdit\_surname->text()==""||ui->lineEdit\_name->text()==""||ui->lineEdit\_age->text()==""||ui->lineEdit\_weight->text()=="")

{

QMessageBox msgBox;

msgBox.setWindowTitle("Сообщение");

msgBox.setText("Заполните все поля!");

msgBox.*exec*();

}

else

{

int count=0;

QString A,B,C,D,E;

A=ui->lineEdit\_surname->text();

B=ui->lineEdit\_name->text();

C=ui->lineEdit\_age->text();

D=ui->lineEdit\_weight->text();

E=ui->comboBox->currentText();

for (int i=0;i<a2;i++)

{

QString A2,B2,C2,D2,E2;

A2=ui->tableWidget->item(i,0)->text();

B2=ui->tableWidget->item(i,1)->text();

C2=ui->tableWidget->item(i,2)->text();

D2=ui->tableWidget->item(i,3)->text();

E2=ui->tableWidget->item(i,4)->text();

if (A==A2 && B==B2 && C==C2 && D==D2 &&E==E2)

{

count++;

}

}

if (count>0)

{

QMessageBox::warning(this,"Ошибка","такой эемент уже есть!");

count=0;

}

else

{

a2++;

ui->tableWidget->setRowCount(a2);

ui->tableWidget->setItem(b2,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(b2,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(b2,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(b2,3,new QTableWidgetItem(D));

ui->tableWidget->setItem(b2,4,new QTableWidgetItem(E));

b2++;

}

}

}

selection1=-1;

}

void addcardio::**on\_pushButton\_edit\_clicked**()

{

if (ui->lineEdit\_edit->text().toUInt()<=a2)

{

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,0)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,1)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,2)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,3)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,4)->setBackground(Qt::green);

selection1=ui->lineEdit\_edit->text().toInt()-1;

}

}

void addcardio::**on\_pushButton\_edit\_2\_clicked**()

{

this->close();

}

void addcardio::**on\_pushButton\_delete\_clicked**()

{

for(int i=0;i<=a2-1;i++)

{

if (i==ui->lineEdit\_delete->text().toInt()-1)

{

for (int j=i;j<a2-1;j++)

{

QString A,B,C,D,E;

A=ui->tableWidget->item(j+1,0)->text();

B=ui->tableWidget->item(j+1,1)->text();

C=ui->tableWidget->item(j+1,2)->text();

D=ui->tableWidget->item(j+1,3)->text();

E=ui->tableWidget->item(j+1,4)->text();

ui->tableWidget->setItem(j,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(j,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(j,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(j,3,new QTableWidgetItem(D));

ui->tableWidget->setItem(j,4,new QTableWidgetItem(E));

}

a2--;

b2--;

count\_del1++;

ui->tableWidget->setRowCount(a2);

}

}

}

# ПРИЛОЖЕНИЕ F

# (обязательное)

# ЛИСТИНГ ФАЙЛА “addcardio.h”

#ifndef ADDCARDIO\_H

#define ADDCARDIO\_H

#include <QDialog>

namespace **Ui** {

class **addcardio**;

}

class **addcardio** : public QDialog

{

Q\_OBJECT

public:

explicit **addcardio**(QWidget \*parent = nullptr);

~***addcardio***();

void **tcardio**(QString &A,QString &B,QString &C, QString &D, QString &E, int l2);

void **set\_num2** (int &num);

void **set2**(QString &A,QString &B,QString &C, QString &D,QString &E);

void **zero2**();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_edit\_clicked**();

void **on\_pushButton\_edit\_2\_clicked**();

void **on\_pushButton\_delete\_clicked**();

private:

Ui::addcardio \*ui;

};

#endif // ADDCARDIO\_H

# ПРИЛОЖЕНИЕ G

# (обязательное)

# ЛИСТИНГ ФАЙЛА “addpower.cpp”

#include "addpower.h"

#include "ui\_addpower.h"

#include <QMessageBox>

#include <QDebug>

#include <iostream>

#include <QTextCodec>

using namespace std;

addpower::**addpower**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::addpower)

{

ui->setupUi(this);

QRegExp Const\_v ("^[А-Яа-я]{,50}$");

QRegExpValidator \*const\_all = new QRegExpValidator(Const\_v, this);

ui->lineEdit\_surname->setValidator(const\_all);

ui->lineEdit\_name->setValidator(const\_all);

QRegExp Const\_dd ("^[0-9]+$");

QRegExpValidator \*const\_delete = new QRegExpValidator(Const\_dd, this);

ui->lineEdit\_delete->setValidator(const\_delete);

ui->lineEdit\_edit->setValidator(const\_delete);

}

addpower::~***addpower***()

{

delete ui;

}

int a=0;

int b=0;

int selection=-1;

int count\_del=0;

void addpower::**tpower**(QString &A, QString &B, QString &C, QString &D, int l)

{

A=ui->tableWidget->item(l,0)->text();

B=ui->tableWidget->item(l,1)->text();

C=ui->tableWidget->item(l,2)->text();

D=ui->tableWidget->item(l,3)->text();

}

void addpower::**set\_num**(int &num)

{

num=a;

}

void addpower::**set**(QString &A, QString &B, QString &C, QString &D)

{

a++;

ui->tableWidget->setRowCount(a);

ui->tableWidget->setItem(b,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(b,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(b,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(b,3,new QTableWidgetItem(D));

b++;

}

void addpower::**zero**()

{

a=0;b=0;count\_del=0;

}

void addpower::**on\_pushButton\_clicked**()

{

if(selection>-1)

{

QTextCodec::setCodecForLocale( QTextCodec::codecForName( "UTF-8" ) );

if(ui->lineEdit\_surname->text()==""||ui->lineEdit\_name->text()==""||(!ui->radioButton->isChecked()&& !ui->radioButton\_2->isChecked())||(!ui->checkBox->isChecked() && !ui->checkBox\_2->isChecked())||(ui->checkBox->isChecked() && ui->checkBox\_2->isChecked()))

{

QMessageBox msgBox;

msgBox.setWindowTitle("Сообщение");

msgBox.setText("Заполните все поля!");

msgBox.*exec*();

}

else

{

QString A,B,C,D;

A=ui->lineEdit\_surname->text();

B=ui->lineEdit\_name->text();

if(ui->radioButton->isChecked()){

C=ui->radioButton->text();

}

if(ui->radioButton\_2->isChecked()){

C=ui->radioButton\_2->text();

}

if(ui->checkBox->isChecked()){

D=ui->checkBox->text();

}

if(ui->checkBox\_2->isChecked()){

D=ui->checkBox\_2->text();

}

ui->tableWidget->setItem(selection,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(selection,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(selection,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(selection,3,new QTableWidgetItem(D));

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,0)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,1)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,2)->setBackground(Qt::white);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,3)->setBackground(Qt::white);

}

}

else

{

if(ui->lineEdit\_surname->text()==""||ui->lineEdit\_name->text()==""||(!ui->radioButton->isChecked()&& !ui->radioButton\_2->isChecked())||(!ui->checkBox->isChecked() && !ui->checkBox\_2->isChecked())||(ui->checkBox->isChecked() && ui->checkBox\_2->isChecked()))

{

QMessageBox msgBox;

msgBox.setWindowTitle("Сообщение");

msgBox.setText("Заполните все поля!");

msgBox.*exec*();

}

else

{

int count=0;

QString A,B,C,D;

A=ui->lineEdit\_surname->text();

B=ui->lineEdit\_name->text();

if(ui->radioButton->isChecked()){

C=ui->radioButton->text();

}

if(ui->radioButton\_2->isChecked()){

C=ui->radioButton\_2->text();

}

if(ui->checkBox->isChecked()){

D=ui->checkBox->text();

}

if(ui->checkBox\_2->isChecked()){

D=ui->checkBox\_2->text();

}

for (int i=0;i<a;i++)

{

QString A2,B2,C2,D2;

A2=ui->tableWidget->item(i,0)->text();

B2=ui->tableWidget->item(i,1)->text();

C2=ui->tableWidget->item(i,2)->text();

D2=ui->tableWidget->item(i,3)->text();

if (A==A2 && B==B2 && C==C2 && D==D2)

{

count++;

}

}

if (count>0)

{

QMessageBox::warning(this,"Ошибка","такой эемент уже есть!");

count=0;

}

else

{

a++;

ui->tableWidget->setRowCount(a);

ui->tableWidget->setItem(b,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(b,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(b,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(b,3,new QTableWidgetItem(D));

b++;

}

}

}

selection=-1;

}

void addpower::**on\_pushButton\_edit\_clicked**()

{

if (ui->lineEdit\_edit->text().toUInt()<=a)

{

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,0)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,1)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,2)->setBackground(Qt::green);

ui->tableWidget->item(ui->lineEdit\_edit->text().toInt()-1,3)->setBackground(Qt::green);

selection=ui->lineEdit\_edit->text().toInt()-1;

}

}

void addpower::**on\_pushButton\_edit\_2\_clicked**()

{

this->close();

}

void addpower::**on\_pushButton\_delete\_clicked**()

{

for(int i=0;i<=a-1;i++)

{

if (i==ui->lineEdit\_delete->text().toInt()-1)

{

for (int j=i;j<a-1;j++)

{

QString A,B,C,D;

A=ui->tableWidget->item(j+1,0)->text();

B=ui->tableWidget->item(j+1,1)->text();

C=ui->tableWidget->item(j+1,2)->text();

D=ui->tableWidget->item(j+1,3)->text();

ui->tableWidget->setItem(j,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(j,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(j,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(j,3,new QTableWidgetItem(D));

}

a--;

b--;

count\_del++;

ui->tableWidget->setRowCount(a);

}

}

}

# ПРИЛОЖЕНИЕ H

# (обязательное)

# ЛИСТИНГ ФАЙЛА “addpower.h”

#ifndef ADDPOWER\_H

#define ADDPOWER\_H

#include <QDialog>

namespace **Ui** {

class **addpower**;

}

class **addpower** : public QDialog

{

Q\_OBJECT

public:

explicit **addpower**(QWidget \*parent = nullptr);

~***addpower***();

void **tpower**(QString &A,QString &B,QString &C, QString &D, int l);

void **set\_num** (int &num);

void **set**(QString &A,QString &B,QString &C, QString &D);

void **zero**();

private slots:

void **on\_pushButton\_clicked**();

void **on\_pushButton\_edit\_clicked**();

void **on\_pushButton\_edit\_2\_clicked**();

void **on\_pushButton\_delete\_clicked**();

private:

Ui::addpower \*ui;

};

#endif // ADDPOWER\_H

# ПРИЛОЖЕНИЕ I

# (обязательное)

# ЛИСТИНГ ФАЙЛА “client1.cpp”

#ifndef CLIENT1\_H

#define CLIENT1\_H

class **client1**

{

public:

**client1**();

};

#include <QString>

class **Cl** {

QString name;

QString surname;

QString type;

public:

**Cl**();

**Cl**(QString name1, QString surname1, QString type1): name(name1), surname(surname1), type(type1) {}

void **gettype**(QString &A)

{

A=type;

}

void **setInfo**(QString &A, QString &B)

{

A=name;

B=surname;

}

void **getname** (QString &A){A=name;}

void **getsurname** (QString &A){A=surname;}

};

class **power** : public Cl

{

QString gender;

QString massage;

public:

**power**();

**power**(QString name1, QString surname1, QString type1, QString gender1, QString massage1): Cl(name1, surname1, type1), gender(gender1), massage(massage1) {}

void **set\_power**(QString &A,QString &B)

{

A=gender;

B=massage;

}

};

class **cardio** : public Cl

{

QString age;

QString weight;

QString time;

public:

**cardio**();

**cardio**(QString name1, QString surname1, QString type1, QString age1, QString weight1, QString time1): Cl(name1, surname1, type1), age(age1), weight(weight1), time(time1) {}

void **set\_cardio**(QString &A,QString &B,QString &C)

{

A=age;

B=weight;

C=time;

}

};

class **aqua** : public Cl

{

QString reference;

public:

**aqua**();

**aqua**(QString name1, QString surname1, QString type1, QString reference1): Cl(name1, surname1, type1), reference(reference1) {}

void **set\_aqua**(QString &A)

{

A=reference;

}

};

#endif // CLIENT1\_H

# ПРИЛОЖЕНИЕ J

# (обязательное)

# ЛИСТИНГ ФАЙЛА “client1.h”

#ifndef CLIENT1\_H

#define CLIENT1\_H

class **client1**

{

public:

**client1**();

};

#include <QString>

class **Cl** {

QString name;

QString surname;

QString type;

public:

**Cl**();

**Cl**(QString name1, QString surname1, QString type1): name(name1), surname(surname1), type(type1) {}

void **gettype**(QString &A)

{

A=type;

}

void **setInfo**(QString &A, QString &B)

{

A=name;

B=surname;

}

void **getname** (QString &A){A=name;}

void **getsurname** (QString &A){A=surname;}

};

class **power** : public Cl

{

QString gender;

QString massage;

public:

**power**();

**power**(QString name1, QString surname1, QString type1, QString gender1, QString massage1): Cl(name1, surname1, type1), gender(gender1), massage(massage1) {}

void **set\_power**(QString &A,QString &B)

{

A=gender;

B=massage;

}

};

class **cardio** : public Cl

{

QString age;

QString weight;

QString time;

public:

**cardio**();

**cardio**(QString name1, QString surname1, QString type1, QString age1, QString weight1, QString time1): Cl(name1, surname1, type1), age(age1), weight(weight1), time(time1) {}

void **set\_cardio**(QString &A,QString &B,QString &C)

{

A=age;

B=weight;

C=time;

}

};

class **aqua** : public Cl

{

QString reference;

public:

**aqua**();

**aqua**(QString name1, QString surname1, QString type1, QString reference1): Cl(name1, surname1, type1), reference(reference1) {}

void **set\_aqua**(QString &A)

{

A=reference;

}

};

#endif // CLIENT1\_H

# ПРИЛОЖЕНИЕ K

# (обязательное)

# ЛИСТИНГ ФАЙЛА “show\_all.cpp”

#include "show\_all.h"

#include "ui\_show\_all.h"

show\_all::**show\_all**(QWidget \*parent) :

QDialog(parent),

ui(new Ui::show\_all)

{

ui->setupUi(this);

}

show\_all::~***show\_all***()

{

delete ui;

}

int j=0,p=0;

int j2=0,p2=0;

int j3=0,p3=0;

void show\_all::**on\_pushButton\_edit\_2\_clicked**()

{

this->close();

}

void show\_all::**get**(QString &A, QString &B, QString &C, QString &D)

{

p++;

ui->tableWidget->setRowCount(p);

ui->tableWidget->setItem(j,0,new QTableWidgetItem(A));

ui->tableWidget->setItem(j,1,new QTableWidgetItem(B));

ui->tableWidget->setItem(j,2,new QTableWidgetItem(C));

ui->tableWidget->setItem(j,3,new QTableWidgetItem(D));

j++;

}

void show\_all::**get2**(QString &A, QString &B, QString &C, QString &D, QString &E)

{

p2++;

ui->tableWidget\_2->setRowCount(p2);

ui->tableWidget\_2->setItem(j2,0,new QTableWidgetItem(A));

ui->tableWidget\_2->setItem(j2,1,new QTableWidgetItem(B));

ui->tableWidget\_2->setItem(j2,2,new QTableWidgetItem(C));

ui->tableWidget\_2->setItem(j2,3,new QTableWidgetItem(D));

ui->tableWidget\_2->setItem(j2,4,new QTableWidgetItem(E));

j2++;

}

void show\_all::**get3**(QString &A, QString &B, QString &C)

{

p3++;

ui->tableWidget\_3->setRowCount(p3);

ui->tableWidget\_3->setItem(j3,0,new QTableWidgetItem(A));

ui->tableWidget\_3->setItem(j3,1,new QTableWidgetItem(B));

ui->tableWidget\_3->setItem(j3,2,new QTableWidgetItem(C));

j3++;

}

void show\_all::**zero**()

{

j=0,p=0;

j2=0,p2=0;

j3=0,p3=0;}

# ПРИЛОЖЕНИЕ L

# (обязательное)

# ЛИСТИНГ ФАЙЛА “show\_all.h”

#ifndef SHOW\_ALL\_H

#define SHOW\_ALL\_H

#include <QDialog>

namespace **Ui** {

class **show\_all**;

}

class **show\_all** : public QDialog

{

Q\_OBJECT

public:

explicit **show\_all**(QWidget \*parent = nullptr);

~***show\_all***();

void **get**(QString &A,QString &B,QString &C,QString &D);

void **get2**(QString &A,QString &B,QString &C,QString &D,QString &E);

void **get3**(QString &A,QString &B,QString &C);

void **zero**();

private slots:

void **on\_pushButton\_edit\_2\_clicked**();

private:

Ui::show\_all \*ui;

};

#endif // SHOW\_ALL\_H