Содержание

[Введение 5](#_Toc74257339)

[Раздел 1 Предпроектное обследование 6](#_Toc74257340)

[1.1 Исследование предметной области 6](#_Toc74257341)

[1.2 Формирование требований к программному изделию 8](#_Toc74257342)

[Раздел 2 Техническое задание 10](#_Toc74257343)

[2.1 Постановка задачи 10](#_Toc74257344)

[2.2 Концепция информационной базы и её структура 11](#_Toc74257345)

[2.3 Неформальное описание алгоритма работы программного продукта 12](#_Toc74257346)

[2.4 Требования к контролю, обновлению и восстановлению данных, защите данных от разрушений 18](#_Toc74257347)

[2.5 Функции подсистем (модулей) 20](#_Toc74257348)

[2.6 Оценка экономической эффективности от внедрения программы 23](#_Toc74257349)

[Раздел 3 Эскизный проект 25](#_Toc74257350)

[Раздел 4 Разработка проектных решений 37](#_Toc74257351)

[Заключение 38](#_Toc74257352)

[Список использованных источников 39](#_Toc74257353)

[Приложение А Алгоритм формы авторизации 41](#_Toc74257354)

[Приложение Б Алгоритм работы вывода таблиц 42](#_Toc74257356)

[Приложение В Алгоритм работы вкладки «Запросы» 43](#_Toc74257358)

[Приложение Г Алгоритм добавления записи проживающего 44](#_Toc74257360)

[Приложение Д Алгоритм добавления записи данных о родственнике 45](#_Toc74257362)

[Приложение Е Алгоритм добавления записи медицинского работника 46](#_Toc74257364)

[Приложение Ж Алгоритм работы удаления записи проживающего 47](#_Toc74257366)

[Приложение И Алгоритм работы удаления данных о родственнике 48](#_Toc74257368)

[Приложение К Алгоритм работы удаления записи медицинского работника 49](#_Toc74257370)

[Приложение Л Алгоритм работы редактирования записи проживающего 50](#_Toc74257372)

[Приложение М Алгоритм работы редактирования данных о родственнике 51](#_Toc74257374)

[Приложение Н Алгоритм работы редактирования записи корпуса 52](#_Toc74257376)

[Приложение П Алгоритм работы редактирования записи медицинского работника 53](#_Toc74257378)

Приложение Р [Алгоритм работы окна «Справка» 54](#_Toc74257381)

[Приложение С Алгоритм работы поиска записи 55](#_Toc74257382)

[Приложение Т Руководство оператора 56](#_Toc74257384)

[Приложение У Руководство программиста 81](#_Toc74257386)

[Приложение Ф Программный код 87](#_Toc74257388)

**Введение**

Тема дипломного проекта: «Разработка приложения для учёта проживающих в КГБСУСО «Шелаболихинский психоневрологический интернат»».

Целью создания программного продукта является упрощения процесса ведения учета проживающих для краевого государственного бюджетного стационарного учреждения социального обслуживания «Шелаболихинский психоневрологический интернат».

Программа позволит учитывать как множество личных данных по проживающему (дата и место рождения, дата поступления в интернат, диагноз, место проживания в интернате – корпус и палата, данные о близких родственниках, данные личных документов и т.д.), так и все данные об медицинских работниках интерната и родственниках проживающих.

Программное изделие будет иметь возможность осуществлять поиск по записям, создавать, удалять и редактировать записи, что позволит работнику интерната сэкономить время при создании документов и ведения учета.

Главным требованием организации для разработки программного изделия стало исключение какой-либо связи с интернет ресурсами.

Программное изделие разрабатывается по заказу КГБСУСО «Шелаболихинский психоневрологический интернат»», чем объясняется актуальность данной темы.

Программный продукт должен работать с СУБД MS SQL Server.

Исходя из совокупности требований, был выбран язык программирования C# и интегрированная среда разработки программного обеспечения Microsoft Visual Studio 2019.

Программный продукт разрабатывается под операционные системы Windows 7 и выше.

# Раздел 1 Предпроектное обследование

## Исследование предметной области

Основным видом деятельности организация КГБСУСО «Шелаболихинский психоневрологический интернат» является уход за престарелыми и инвалидами с обеспечением проживания.

Заместитель директора по медицинской части ведет учет проживающих для возможной сдачи отчетности по этой информации при необходимости.

По требованию заказчика работа создаваемого программного продукта должна производиться без связи с интернет ресурсами для обеспечения стабильности соединения и быстрого обмена данными.

При исследовании предметной области были выявлены следующие термины:

Психоневрологический интернат – стационарное учреждение для социального обслуживания лиц, страдающих психическими расстройствами, утративших частично или полностью способность к самообслуживанию и нуждающихся по состоянию психического, а нередко и физического здоровья в постоянном уходе и наблюдении.

Проживающие – лица, которые на постоянной основе проживают в психоневрологическом интернате (пожилые люди, инвалиды старше 18 лет, страдающие хроническими психическими расстройствами и нуждающиеся в постороннем уходе).

Программный продукт в ходе работы должен выполнять следующие функции:

* создание, редактирование и удаление записей таблицы «Проживающие»;
* создание, редактирование и удаление записей таблицы «Данные о родственниках»;
* редактирование записей таблицы «Корпуса»;
* создание, редактирование и удаление записей таблицы «Медицинские работники»;
* поиск по записям таблицы «Проживающие»;
* поиск по записям таблицы «Данные о родственниках»;
* поиск по записям таблицы «Корпуса»;
* поиск по записям таблицы «Медицинские работники»;
* вывод таблицы «Проживающие» с фильтрацией по диагнозу.

## 1.2 Формирование требований к программному изделию

Оформление программного продукта должно быть простым и не включать в себя излишней нагрузки на пользователя. При создании программы должны использоваться простые стилистические приемы:

Фон – должен быть однотонный, не яркий для избегания нагрузки на глаза.

Цвета (внутри программного изделия) – должны быть логически оправданы по своему применению в программном изделии (при их использовании).

Кнопки, текстовые поля, панели с инструментами в программном изделии должны находиться рядом друг с другом, соответствуя их использованию.

Справочный материал по использованию программного изделия не должен перегружать смысловой нагрузкой пользователя.

Программное изделие в ходе работы должно выполнять следующие функции:

* создание, редактирование и удаление записей таблицы «Проживающие»;
* создание, редактирование и удаление записей таблицы «Данные о родственниках»;
* редактирование записей таблицы «Корпуса»;
* создание, редактирование и удаление записей таблицы «Медицинские работники»;
* поиск по записям таблицы «Проживающие»;
* поиск по записям таблицы «Данные о родственниках»;
* поиск по записям таблицы «Корпуса»;
* поиск по записям таблицы «Медицинские работники»;
* вывод таблицы «Проживающие» с фильтрацией по диагнозу.

Время выполнения операций не должно превышать 40 секунд с момента начала обработки информации программным изделием.

Программный продукт будет использоваться организацией КГБСУСО «Шелаболихинский психоневрологический интернат» и должно обмениваться данными с Microsoft SQL Server.

# Раздел 2 Техническое задание

## 2.1 Постановка задачи

Главной задачей программного изделия является работа с базой данных. Для вывода таблиц из базы данных в приложение используется элемент DataGridView, который позволяет вывести данные в реляционном виде.

Вывод проживающих с фильтрацией по диагнозу будет также произведен в DataGridView.

Для создания новой записи проживающего в базу данных будет разработана форма, содержащая такие поля как: ФИО, Пол, Дата рождения, Дата поступления, Диагноз, Номер корпуса, Номер палаты, Номер паспорта, Серия паспорта, Место рождения и Номер телефона (при наличии). Для осуществления добавления нужно будет заполнить все существующие тестовые поля и нажать на кнопку «Добавить».

Для создания новой записи медицинского работника в базу данных будет разработана форма, содержащая такие поля как: ФИО, Пол, Дата рождения, Должность, Код корпуса, Номер телефона, Номер паспорта, Серия паспорта, Место рождения, Место жительства. Для осуществления добавления нужно будет заполнить все существующие тестовые поля и нажать на кнопку «Добавить».

Для создания новой записи данных о родственнике в базу данных будет разработана форма, содержащая такие поля как: ФИО, Место жительства, Номер телефона, Электронная почта, Проживающий. Для осуществления добавления нужно будет заполнить все существующие тестовые поля и нажать на кнопку «Добавить».

Удаление и редактирование существующих записей в отдельных формах при помощи элемента DataGridView, который показывает все записи таблиц базы данных.

Поиск по записям таблиц будет производиться на главном окне программы с помощью поля textbox.

## 2.2 Концепция информационной базы и её структура

Информационная база включает в себя следующие сущности:

* Проживающие (Код проживающего, ФИО, Пол, Дата рождения, Дата поступления, Диагноз, Код корпуса, Номер палаты, Номер паспорта, Серия паспорта, Место рождения, Номер телефона).
* Данные о родственниках (Код родственника, ФИО, Место жительства, Номер телефона, Электронная почта, Код проживающего).
* Корпуса (Код корпуса, Адрес, Количество палат).
* Медицинские работники (Код медицинского работника, ФИО, Пол, Дата рождения, Должность, Код корпуса, Номер телефона, Номер паспорта, Серия паспорта, Место рождения, Место жительства).

Инфологическая модель указана на рисунке 1.

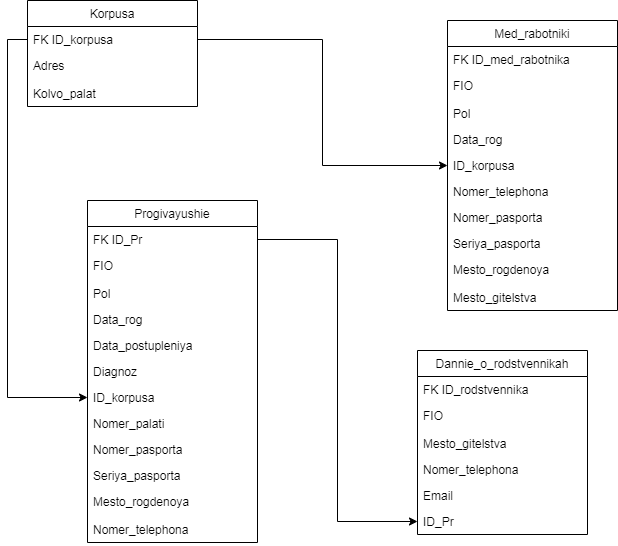


Рисунок 1 – Инфологическая модель задачи

## 2.3 Неформальное описание алгоритма работы программного продукта

После запуска программы открывается окно авторизации, на котором расположено два поля – поле для ввода логина и поле для ввода пароля, а так же кнопка «Войти». После успешного ввода логина и пароля пользователь переходит на главное окно программы, если же логин и/или пароль введен(ы) неверно пользователь получит сообщение об ошибке.

Главное окно содержит меню с вкладками «Таблицы», «Запросы», «Ввод данных» и кнопками «Удаление данных», «Редактирование данных» «Справка» и «Сменить пользователя».

При нажатии на вкладку «Таблицы» перед пользователем появится список таблиц базы данных (таблицы «Проживающие», «Данные о родственниках», «Корпуса», «Медицинские работники»).

При нажатии на вкладку «Запросы» появится список для вывода проживающих с фильтрацией по диагнозу.

При нажатии на кнопку «Проживающие» во вкладке «Таблицы» появляется таблица, содержащая данные о проживающих из базы данных.

Для добавления новой записи о проживающем необходимо нажать на вкладку «Ввод данных» и затем на кнопку «Добавление данных о проживающем». Перед пользователем появится новое окно «Добавление проживающего». Для добавления новой записи пользователь должен заполнить все текстовые поля и нажать кнопку добавить. Если все поля заполнены верно, пользователь получит сообщение «Обработана 1 запись», если же было пропущено одно из полей, либо же какое то поле было заполнено неправильно программа выдаст сообщение об ошибке и данные не будут добавлены в базу данных. Для очистки всех полей ввода существует кнопка «Очистить поля», при нажатии на которую все поля, которые пользователь заполнил, станут пустыми.

Для удаления записи из таблицы «Проживающие» необходимо нажать на вкладку «Удаление данных». Перед пользователем появится новое окно «Удаление данных» содержащее меню со всеми таблицами базы данных, элемент управления для вывода таблиц, поле для поиска записей, кнопку «Удалить» и кнопку «Назад». Для того чтобы осуществить удаление нужной записи пользователь должен нажать на вкладку «Таблицы» и затем на таблицу «Проживающие». В соответствующем элементе управления появится данная таблица. Когда нужная запись найдена, пользователь должен нажать на нее (выделить строку) и затем нажать на кнопку «Удалить запись». Если все прошло успешно пользователь получит сообщение «Обработана 1 запись», напротив – сообщение об ошибке. Для возврата на главное окно программы пользователь должен нажать на кнопку «Назад».

Для редактирования записи таблицы «Проживающие» пользователю необходимо нажать на вкладку «Редактирование данных» на главной форме программы. Перед пользователем откроется окно «Редактирование данных» содержащее меню со всеми таблицами базы данных, элемент управления для вывода таблиц, кнопку «Сохранить» и кнопку «Назад». Для осуществления редактирования данных о проживающих необходимо нажать на таблицу «Проживающие» из вкладки «Таблицы» и выбрать нужную запись из общего списка двойным нажатием левой кнопки мыши, отредактировать нужные поля и нажать на кнопку сохранения. Для возврата на главное окно программы пользователь должен нажать на кнопку «Назад».

При нажатии на кнопку «Данные о родственниках» во вкладке «Таблицы» появляется таблица, содержащая данные о родственниках проживающих из базы данных.

Для добавления новой записи о родственнике необходимо нажать на вкладку «Ввод данных» и затем на кнопку «Добавление данных о родственниках». Перед пользователем появится новое окно «Добавление данных о родственниках». Для добавления новой записи пользователь должен заполнить все текстовые поля и нажать кнопку добавить. Если все поля заполнены верно, пользователь получит сообщение «Обработана 1 запись», если же было пропущено одно из полей, либо же какое то поле было заполнено неправильно программа выдаст сообщение об ошибке и данные не будут добавлены в базу данных. Для очистки всех полей ввода существует кнопка «Очистить поля», при нажатии на которую все поля, которые пользователь заполнил, станут пустыми.

Для удаления записи из таблицы «Данные о родственниках» необходимо нажать на вкладку «Удаление данных». Перед пользователем появится новое окно «Удаление данных» содержащее меню со всеми таблицами базы данных, элемент управления для вывода таблиц, поле для поиска записей, кнопку «Удалить» и кнопку «Назад». Для того чтобы осуществить удаление нужной записи пользователь должен нажать на вкладку «Таблицы» и затем на таблицу «Данные о родственниках». В соответствующем элементе управления появится данная таблица. Когда нужная запись найдена, пользователь должен нажать на нее (выделить строку) и затем нажать на кнопку «Удалить запись». Если все прошло успешно пользователь получит сообщение «Обработана 1 запись», напротив – сообщение об ошибке. Для возврата на главное окно программы пользователь должен нажать на кнопку «Назад».

Для редактирования записи таблицы «Данные о родственниках» пользователю необходимо нажать на вкладку «Редактирование данных» на главной форме программы. Перед пользователем откроется окно «Редактирование данных» содержащее меню со всеми таблицами базы данных, элемент управления для вывода таблиц, кнопку «Сохранить» и кнопку «Назад». Для осуществления редактирования данных о родственниках необходимо нажать на таблицу «Данные о родственниках» из вкладки «Таблицы» и выбрать нужную запись из общего списка двойным нажатием левой кнопки мыши, отредактировать нужные поля и нажать на кнопку сохранения. Для возврата на главное окно программы пользователь должен нажать на кнопку «Назад».

При нажатии на кнопку «Корпуса» во вкладке «Таблицы» появляется таблица, содержащая данные о корпусах интерната из базы данных.

Для редактирования записи таблицы «Корпуса» пользователю необходимо нажать на вкладку «Редактирование данных» на главной форме программы. Перед пользователем откроется окно «Редактирование данных» содержащее меню со всеми таблицами базы данных, элемент управления для вывода таблиц, кнопку «Сохранить» и кнопку «Назад». Для осуществления редактирования данных о корпусах необходимо нажать на таблицу «Корпуса» из вкладки «Таблицы» и выбрать нужную запись из общего списка двойным нажатием левой кнопки мыши, отредактировать нужные поля и нажать на кнопку сохранения. Для возврата на главное окно программы пользователь должен нажать на кнопку «Назад».

При нажатии на кнопку «Медицинские работники» во вкладке «Таблицы» появляется таблица, содержащая данные о медицинских работниках из базы данных.

Для добавления новой записи о медицинских работниках необходимо нажать на вкладку «Ввод данных» и затем на кнопку «Добавление данных о медицинских работниках». Перед пользователем появится новое окно «Добавление медицинского работника». Для добавления новой записи пользователь должен заполнить все текстовые поля и нажать кнопку добавить. Если все поля заполнены верно, пользователь получит сообщение «Обработана 1 запись», если же было пропущено одно из полей, либо же какое то поле было заполнено неправильно программа выдаст сообщение об ошибке и данные не будут добавлены в базу данных. Для очистки всех полей ввода существует кнопка «Очистить поля», при нажатии на которую все поля, которые пользователь заполнил, станут пустыми.

Для удаления записи из таблицы «Медицинские работники» необходимо нажать на вкладку «Удаление данных». Перед пользователем появится новое окно «Удаление данных» содержащее меню со всеми таблицами базы данных, элемент управления для вывода таблиц, поле для поиска записей, кнопку «Удалить» и кнопку «Назад». Для того чтобы осуществить удаление нужной записи пользователь должен нажать на вкладку «Таблицы» и затем на таблицу «Медицинские работники». В соответствующем элементе управления появится данная таблица. Когда нужная запись найдена, пользователь должен нажать на нее (выделить строку) и затем нажать на кнопку «Удалить запись». Если все прошло успешно пользователь получит сообщение «Обработана 1 запись», напротив – сообщение об ошибке. Для возврата на главное окно программы пользователь должен нажать на кнопку «Назад».

Для редактирования записи таблицы «Медицинские работники» пользователю необходимо нажать на вкладку «Редактирование данных» на главной форме программы. Перед пользователем откроется окно «Редактирование данных» содержащее меню со всеми таблицами базы данных, элемент управления для вывода таблиц, кнопку «Сохранить» и кнопку «Назад». Для осуществления редактирования данных о проживающих необходимо нажать на таблицу «Медицинские работники» из вкладки «Таблицы» и выбрать нужную запись из общего списка двойным нажатием левой кнопки мыши, отредактировать нужные поля и нажать на кнопку сохранения. Для возврата на главное окно программы пользователь должен нажать на кнопку «Назад».

При нажатии на вкладку «Запросы» пользователь может выбрать, по какому диагнозу отфильтровать проживающих. Если пользователь нажмет на определенный запрос, в таблице появятся те проживающие, диагноз которых соответствует запросу.

Для того чтобы найти нужную запись в таблице существует поле для поиска, в которое пользователь вводит нужную ему информацию (ФИО проживающего, ФИО родственника, ФИО медицинского работника или номер корпуса) и в таблице подсвечиваются нужные данные.

При нажатии на вкладку «Справка» перед пользователем откроется окно содержащее справочную информацию – руководство оператора и руководство программиста.

Схема алгоритма работы окна авторизации указана в Приложении А.

Схема алгоритма вывода таблиц указана в приложении Б.

Схема алгоритма работы вкладки «Запросы» указана в приложении В.

Схема алгоритма добавления записи проживающего указана в приложении Г.

Схема алгоритма добавления записи родственника проживающего указана в приложении Д.

Схема алгоритма добавления записи медицинского работника указана в приложении Е.

Схема алгоритма удаления записи проживающего указана в приложении Ж.

Схема алгоритма удаления записи родственника проживающего указана в приложении И.

Схема алгоритма удаления записи медицинского работника указана в приложении К.

Схема алгоритма редактирования записи проживающего указана в приложении Л.

Схема алгоритма редактирования записи родственника проживающего указана в приложении М.

Схема алгоритма редактирования записи корпуса указана в приложении Н.

Схема алгоритма редактирования записи медицинского работника указана в приложении П.

Схема алгоритма работы окна «Справка» указана в приложении Р.

Схема алгоритма работы поиска указана в приложении С.

## 2.4 Требования к контролю, обновлению и восстановлению данных, защите данных от разрушений

Входной информацией является та информация, которую пользователь заполняет в программном продукте, заполняя необходимые поля [6].

Для разрабатываемого программного продукта входной будет служить следующая информация:

* информация о проживающих;
* информация о родственниках проживающих;
* информация о корпусах;
* информация о медицинских работниках.

Выходная информация – результат выполнения программного алгоритма и внесение информации в базу данных или файл, фильтрация данных. Информация, которая несет вывод либо обобщает указанные данные в общем виде или по определенному критерию [6].

Входная и выходная информации контролируются типами данных переменных и сотрудником (администратором), который вводит эту информацию.

Безопасность и защита данных – самый важный момент при создании программного продукта, необходимый для сохранности и целостности данных.

Для предотвращения потери данных следует сделать резервное копирование.

Резервное копирование – процесс создания копии данных на носителе (жёстком диске, дискете и т. д.), предназначенном для восстановления данных в оригинальном или новом месте их расположения в случае их повреждения или разрушения [7].

Виды резервного копирования:

– полное резервное копирование;

– дифференциальное резервное копирование;

– инкрементное резервное копирование;

– клонирование;

– резервное копирование в виде образа;

– резервное копирование в режиме реального времени;

– холодное резервирование;

– горячее резервирование.

## 2.5 Функции подсистем (модулей)

Одним из самых распространённых в настоящее время методов создания программной продукции является структурное программирование. Один из основных принципов структурного программирования – это принцип модульности. Данный принцип заключается в разделении программы на отдельные законченные фрагменты – модули, блоки, которые отлаживаются и тестируются отдельно друг от друга, а так же просты в управлении и создания. В конечном итоге отдельные модули программного продукта могут разрабатываться разными группами программистов, что позволяет экономить большое количество времени и человеческих ресурсов.

Программа, которая создается в соответствие с принципами структурного программирования, должна отвечать следующим требованиям:

– программа разделена на независимые части, называемые модулями;

– модуль – это независимый блок, код (текст) которого физически и логически отделен от кода других модулей;

– модуль выполняет только одну логическую функцию, иначе говоря, должен решать самостоятельную задачу своего уровня по принципу: один программный модуль – одна логическая функция [9].

Работа программного модуля не должна зависеть [9]:

– от того, какому программному модулю предназначены его выходные данные;

– от предыстории вызовов программного модуля;

– размер программного модуля желательно ограничивать одной**-**двумя страницами исходного листинга (50**-**100 строк исходного кода);

– модуль должен иметь только одну входную и одну выходную точку;

– взаимосвязи между модулями устанавливаются по иерархической структуре;

– каждый модуль должен начинаться с комментария, объясняющего его назначение, назначение переменных, передаваемых в модуль и из него, модулей, которые его вызывают, и модулей, которые вызываются из него;

– оператор безусловного перехода или вообще не используется в модуле, или применяется в исключительных случаях только для перехода на выходную точку модуля;

– в тексте модуля необходимо использовать комментарии, в особенности в сложных местах алгоритма;

– идентификаторы переменных и модулей должны быть смысловыми, «говорящими»;

– в одной строке стоит записывать не более одного оператора. Если для записи оператора требуется больше, чем одна строка, то все последующие операторы записываются с отступами;

– желательно не допускать вложенности блоков операторов более трех уровней;

– следует избегать использования языковых конструкций с неочевидной семантикой и программистских «трюков».

Разработка создаваемого программного продукта велась с соблюдением принципов структурного программирования. Каждая кнопка, смена выбранного элемента в таблицах являются отдельными модулями, которые отделены друг от друга физически и логически. Так же для формирования документов разработаны специальные модули, каждый из которых выполняет свою функцию.

Для добавления новой записи в таблицу «Проживающие» на форме реализована кнопка «Добавить», которая отправляет данные из полей формы в таблицу базы данных. Сохранение новых записей в таблицу «Данные о родственниках» и «Медицинские работники» происходи по тому же принципу.

Для удаления записи из таблицы существует форма «Удаление данных» содержащая кнопку «Удалить запись».

Для сохранения внесенных изменений в таблицах базы данных на форме «Редактирование данных» существует кнопка «Сохранить».

Окно «Справка» содержит в себе руководство оператора и руководство программиста, для углубленного освоения работы программного изделия.

Так же главная форма программы содержит кнопку «Сменить пользователя», которая перенаправляет пользователя на окно авторизации.

## 2.6 Оценка экономической эффективности от внедрения программы

Чтобы оценить экономический эффект от внедрения нового программного продукта, следует рассчитать затраты на разработку проекта и эксплуатационные затраты, связанные с обеспечением его нормального функционирования.

Величину затрат на создание программы определяют на основе метода калькуляций. В этом случае затраты определяются расчетом по отдельным статьям расходов и их последующим суммированием.

Балансовая стоимость ЭВМ включает отпускную цену, расходы на транспортировку, монтаж оборудования и его наладку и вычисляется по формуле:

Сб = Срын + Зуст,

где Сб – балансовая стоимость ЭВМ, р.;

Срын – рыночная стоимость компьютера, р./шт.;

Зуст – затраты на доставку и установку компьютера, %.

Отсюда:

Сб = 40 500 р./шт

Сумму годовых амортизационных отчислений определяем по формуле:

Аг = Сб \* Нам,

где Аг – сумма годовых амортизационных отчислений, р.;

Сб – балансовая стоимость компьютера, р./шт.;

Нам – норма амортизации, %.

Агэвм = 40 500 \* 0,25 = 10125 р.

Следовательно, сумма амортизационных отчислений за период создания проекта будет равняться произведению амортизационных отчислений в день на количество дней эксплуатации компьютера и программного обеспечения при создании программы.

Апэвм = 1664 р.

Общая стоимость программного обеспечения, которое будет использоваться для разработки программного продукта, составляет 8000 р. На программное обеспечение, как и на компьютеры, производятся амортизационные отчисления.

Агпо = 9000 \* 0,25 = 2250 р.

Аппо = = 370 р.

К статье «Затраты на потребляемую электроэнергию» относится стоимость потребляемой электроэнергии компьютером за время разработки программы. Стоимость электроэнергии, потребляемой за год, определяется по формуле:

Зэл = Pэвм \* tэвм \* Сэл,

где Pэвм – суммарная мощность ЭВМ, кВт;

tэвм – время работы компьютера, часов;

Сэл – стоимость 1 кВт /ч. электроэнергии, р.

Согласно техническому паспорту ЭВМ, Pэвм = 0,17 кВт.

Стоимость 1 кВт/ч электроэнергии Сэл = 3,5 р.

Рабочий день равен восьми часам, следовательно, стоимость электроэнергии за период работы компьютера во время создания программы будет равна:

Зэл = 0,17 \* 60 \* 8 \* 3,5 = 285,60 р.

Получаем общую стоимость программного обеспечения равную сумме всех критериев:

Собщая = 55195 р.

# Раздел 3 Эскизный проект

Интерфейс – комплекс условий, обеспечивающих взаимодействие объектов в окружающем информационном пространстве [10].

Любая интерактивная система или программа нуждается в интерфейсе, обеспечивающем удобство и простоту работы пользователя. «Дружественность» интерфейса к пользователю обеспечивает эффективность системы в целом. Дружественный интерфейс должен предотвращать ситуации, которые могут закончиться ошибками, предупреждать пользователя о возможных ошибках и способах их предупреждения, предлагать отмену ошибочных действий. Дружественность интерфейса означает обеспечение простоты работы в сочетании с реализацией доступа к широким функциональным возможностям. Один из возможных путей поддержания простоты работы – представление на экране информации, минимально необходимой для выполнения пользователем очередного шага задания. Другой путь – размещение и представление элементов управления на экране с учетом их смыслового значения и логической взаимосвязи. Этот подход ориентируется на ассоциативное мышление пользователя [14].

Благодаря дружественности интерфейса происходит лёгкость и быстрота работы пользователя:

* легкая загрузка большого количества уже настроенных элементов (окон, экранов, рабочих столов и цветовых схем);
* простой и быстрый способ размещения настраиваемого окна в нужное место экрана;
* удобный и функциональный сервис поиска инструмента или группы инструментов.

Эскиз формы авторизации представлен на рисунке 2.

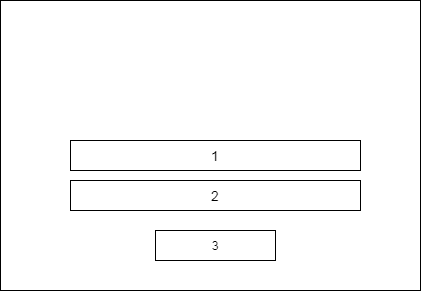


Рисунок 2 – Эскиз формы авторизации

1 – Текстовое поле ввода логина.

2 – Текстовое поле ввода пароля.

3 – Кнопка «Войти».

Эскиз главной формы представлен на рисунке 3.

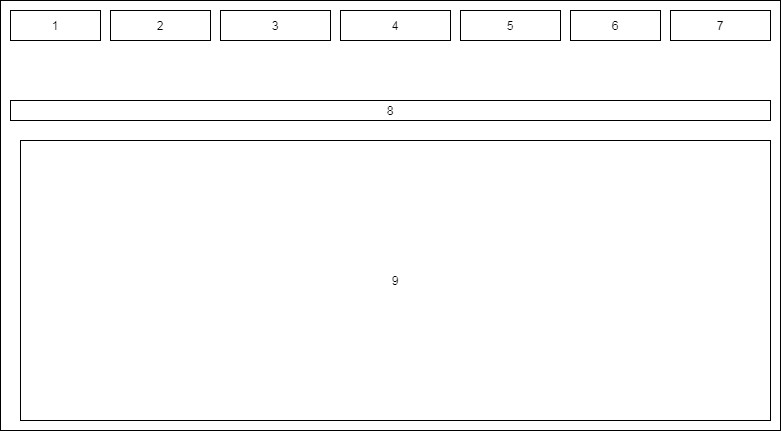


Рисунок 3 – Эскиз главной формы

1 – Вкладка «Таблицы».

2 – Вкладка «Запросы».

3 – Вкладка «Ввод данных».

4 – Кнопка «Удаление данных».

5 – Кнопка «Редактирование данных».

6 – Кнопка «Справка».

7 – Кнопка «Сменить пользователя».

8 – Поле для поиска.

9 – Поле вывода таблиц базы данных.

Эскиз формы добавления проживающего представлен на рисунке 4.

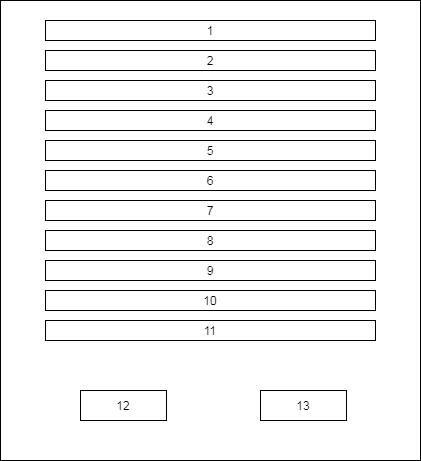


Рисунок 4 – Эскиз формы добавления проживающего

1 – Поле ввода ФИО.

2 – Поле выбора пола.

3 – Поле выбора даты рождения.

4 – Поле выбора даты поступления.

5 – Поле ввода диагноза.

6 – Поле ввода номера корпуса.

7 – Поле ввода номера палаты.

8 – Поле ввода номера паспорта.

9 – Поле вывода серии паспорта.

10 – Поле ввода места рождения.

11 – Поле ввода номера телефона.

12 – Кнопка «Добавить».

13 – Кнопка «Отмена».

Эскиз формы добавления медицинского работника представлен на рисунке 5.

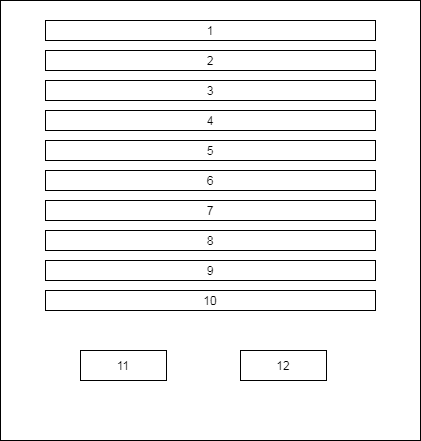


Рисунок 5 – Макет формы добавления медицинского работника

1 – Поле ввода ФИО.

2 – Поле выбора пола.

3 – Поле выбора даты рождения.

4 – Поле ввода должности.

5 – Поле ввода номера корпуса.

7 – Поле ввода номера телефона.

8 – Поле ввода номера паспорта.

9 – Поле вывода серии паспорта.

10 – Поле ввода места рождения.

11 – Поле ввода места жительства.

12 – Кнопка «Добавить».

13 – Кнопка «Отмена».

Эскиз формы добавления данных о родственнике представлен на рисунке 6.

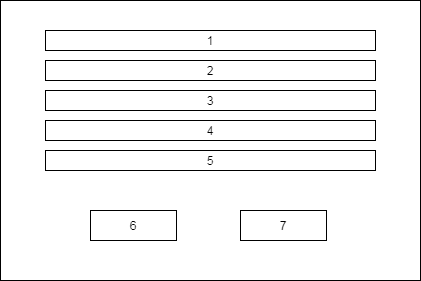


Рисунок 6 – Макет формы добавления данных о родственнике

1 – Поле ввода ФИО.

2 – Поле ввода места жительства.

3 – Поле ввода номера телефона.

4 – Поле ввода электронной почты.

5 – Поле выбора проживающего.

6 – Кнопка «Добавить».

7 – Кнопка «Отмена».

Эскиз формы удаления данных представлен на рисунке 7.

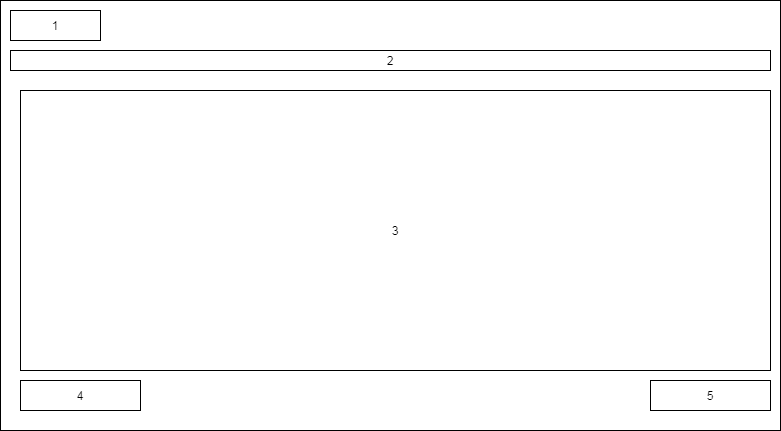


Рисунок 7 – Эскиз формы удаления данных

1 – Вкладка «Таблицы».

2 – Поле для поиска.

3 – Поле вывода таблиц базы данных.

4 – Кнопка «Назад».

5 – Кнопка «Удалить запись».

Эскиз формы редактирования данных представлен на рисунке 8.

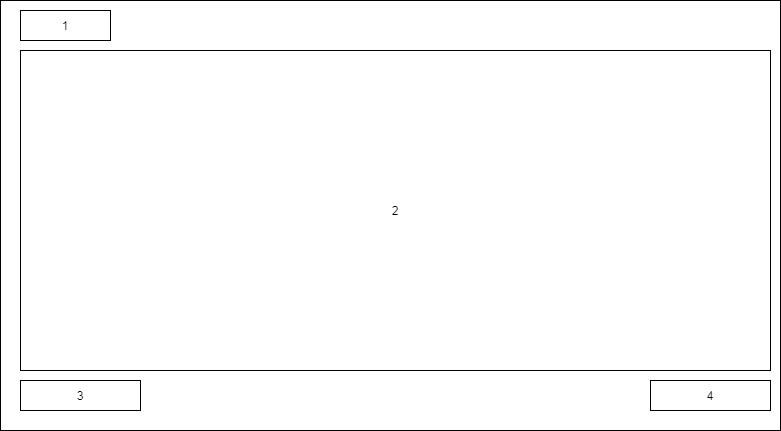


Рисунок 8 – Эскиз формы редактирования данных

1 – Вкладка «Таблицы».

2 – Поле вывода таблиц базы данных.

3 – Кнопка «Назад».

5 – Кнопка «Сохранить».

Эскиз справочной формы представлен на рисунке 9.



Рисунок 9 – Эскиз справочной формы

1 – Вкладка «Руководство оператора».

2 – Вкладка «Руководство программиста».

3 – Поле вывода текстовой информации.

**Контрольный пример**

Первое, что увидит пользователь при запуске программного изделия, это форму авторизации пользователя.

Форма авторизации представлена на рисунке 10.

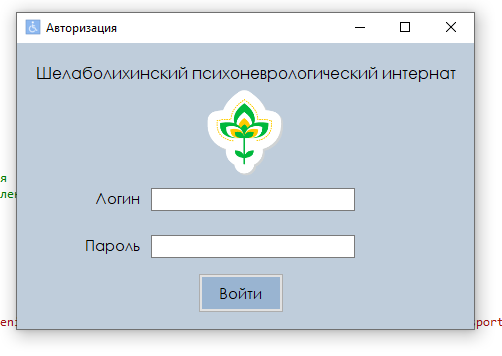


Рисунок 10 – Форма авторизации

После успешной авторизации перед пользователем откроется главная форма программы.

Главная форма программы представлена на рисунке 11.

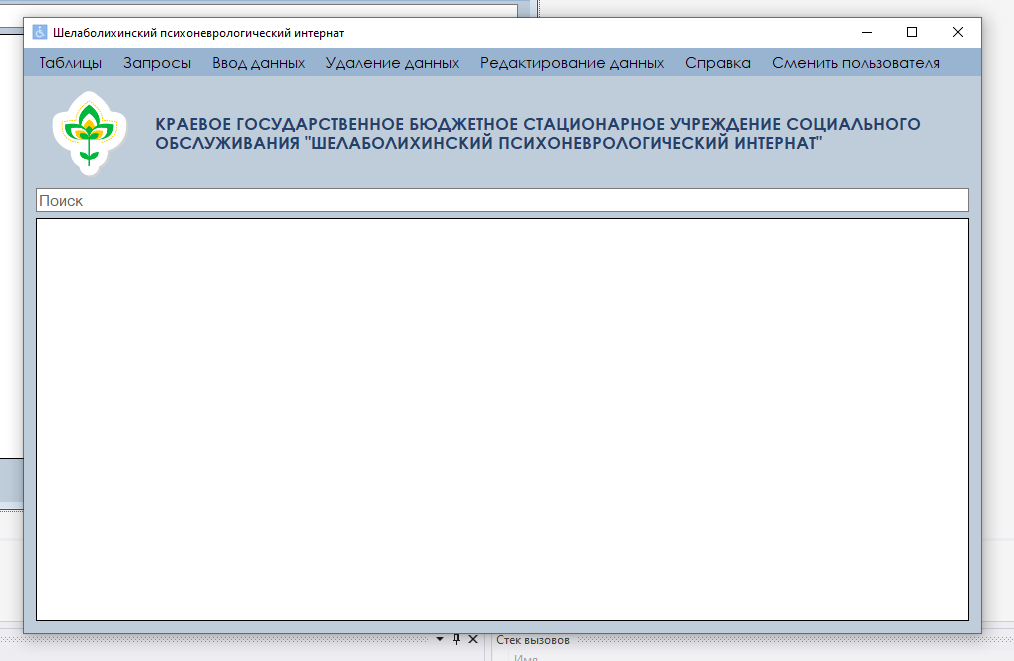


Рисунок 11 – Главная форма программы

При работе с программным изделием в случае ошибки или выполнения тех или иных действий будет выводиться диалоговое окно, которое будет подсказывать пользователю о выполнении требуемой задачи или ошибке при заполнении текстовых полей.

Сообщения пользователю представлены на рисунках 12, 13, 14, 15, 16, 17.

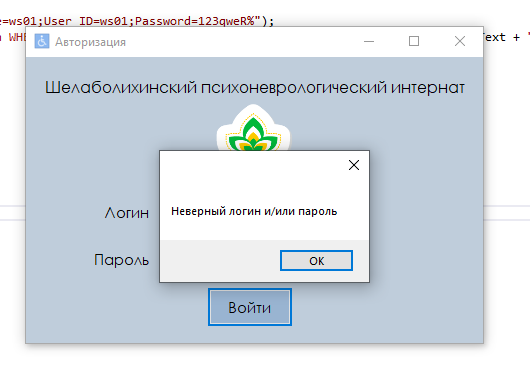


Рисунок 12 – Сообщение об ошибке

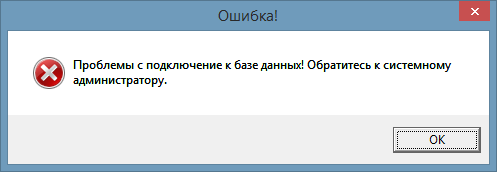


Рисунок 13 – Сообщение об ошибке

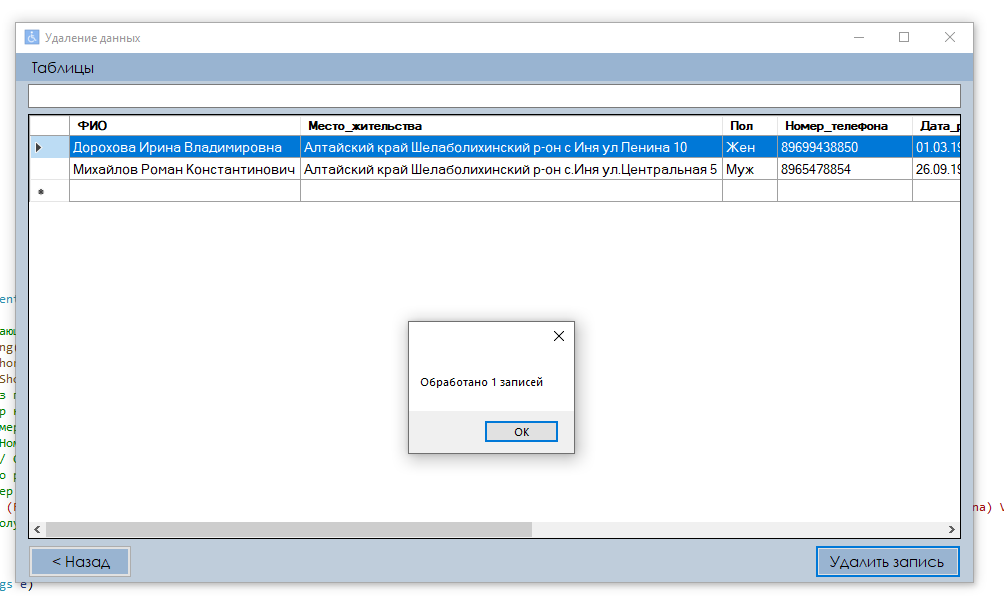


Рисунок 14 – Сообщение о выполнении действия

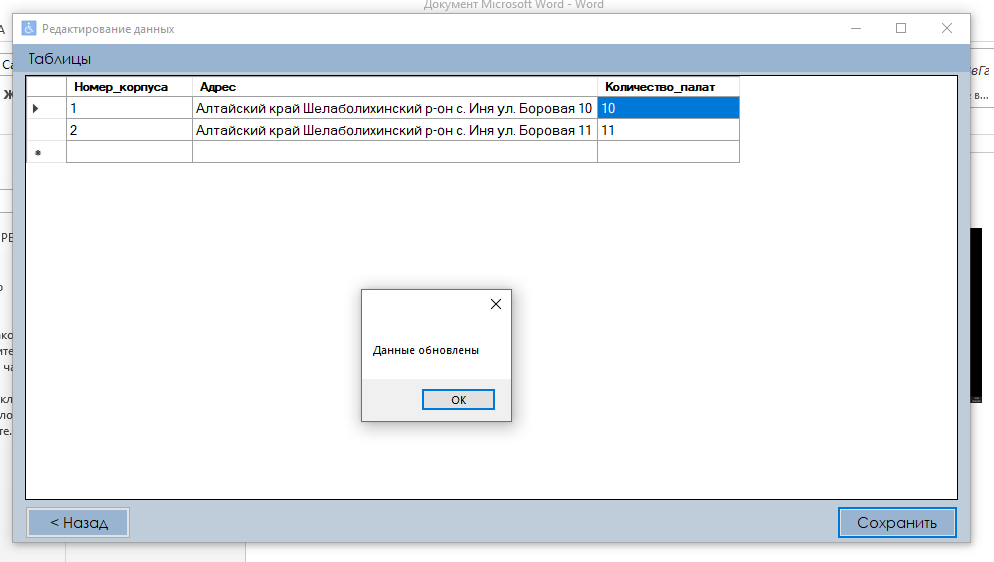


Рисунок 15 – Сообщение о выполнении действия

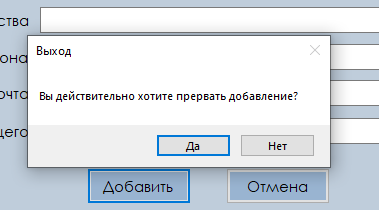


Рисунок 16 – Сообщение о закрытии формы

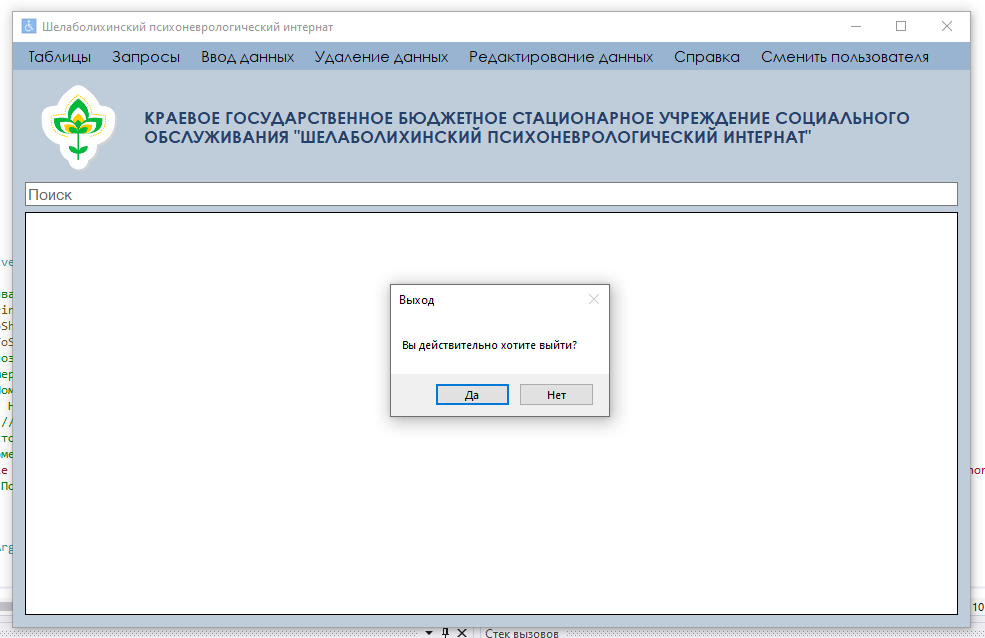


Рисунок 17 – Сообщение о закрытии программы

В помощь пользователю в программном изделии есть кнопка «Справка», которая выводит дополнительную информацию по работе с этим самым изделием.

Окно справки представлено на рисунках 18 и 19.

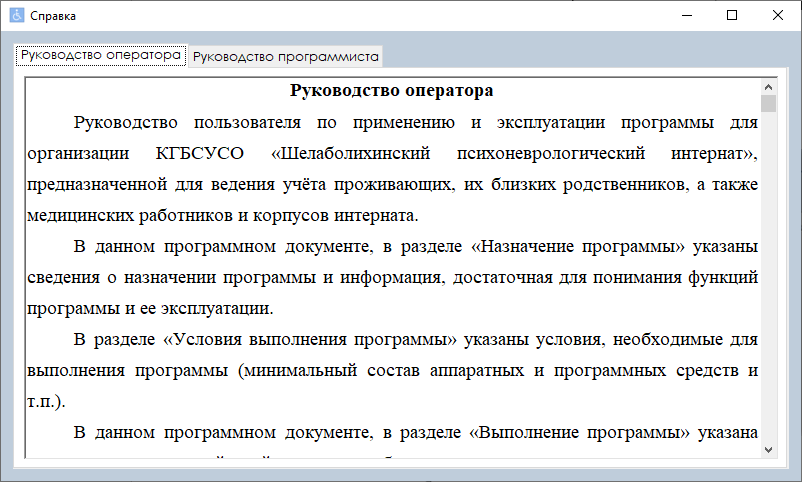


Рисунок 18 – Справочная система

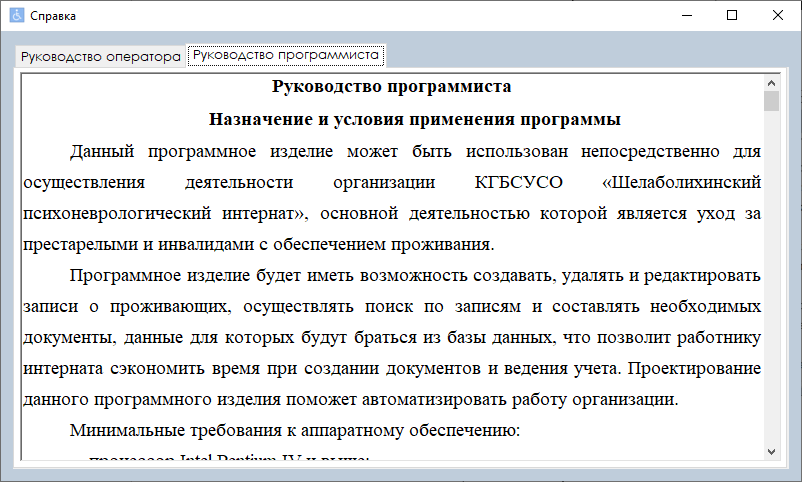


Рисунок 19 – Справочная система

Меню, как и в большинстве программных средств, находится сверху и содержит в себе некоторые функции, которые могут помочь пользователю в работе с программным изделием.

Меню программы представлено на рисунке 20.

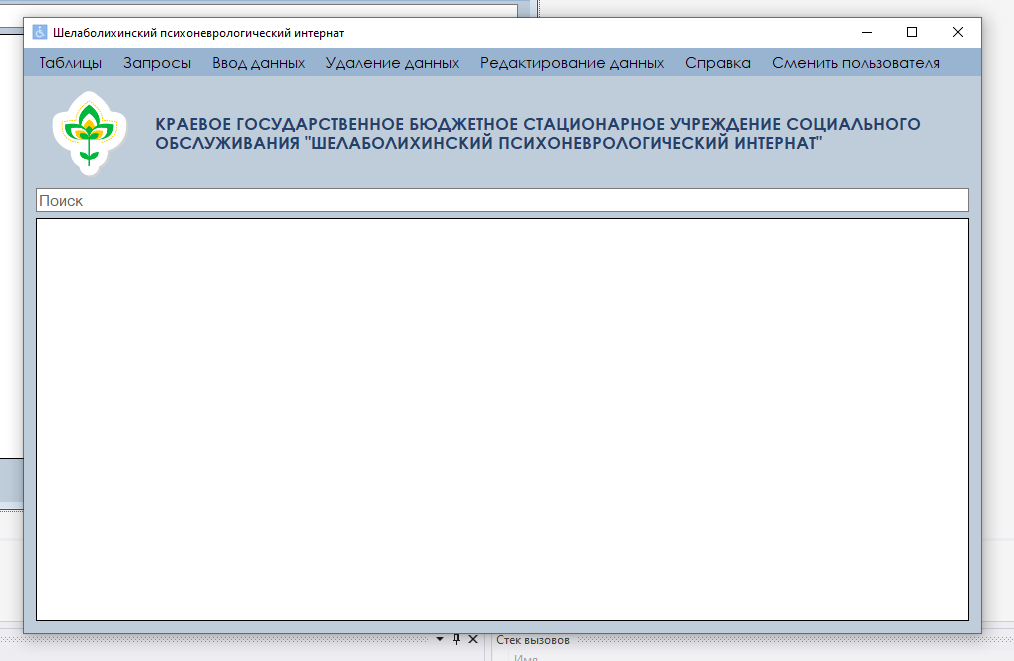


Рисунок 20 – Меню

Кнопки, являющиеся элементами управления работой программного изделия, имеют определённый не выделяющийся цветовой стиль.

Кнопки управления работой программы представлены на рисунках 21, 22, 23, 24, 25, 26.

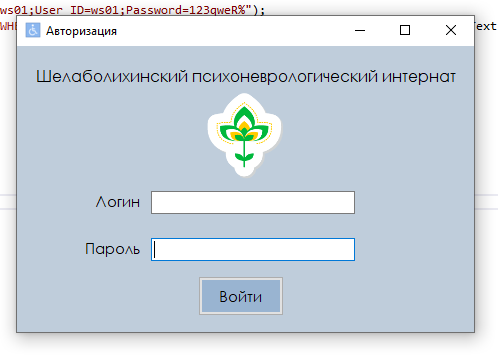


Рисунок 21 – Кнопка «Войти»

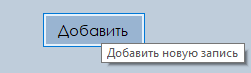


Рисунок 22 – Кнопка «Добавить»

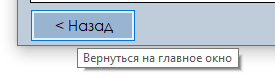


Рисунок 23 – Кнопка «Назад»

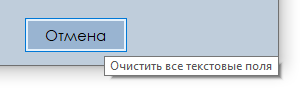


Рисунок 24 – Кнопка «Отмена»

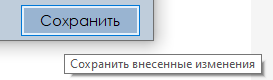


Рисунок 25 – Кнопка «Сохранить»

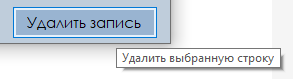


Рисунок 26 – Кнопка «Удалить запись»

Под правильностью алгоритма понимается соответствие результатов работы алгоритма условию задачи, а именно программа должна не просто выдавать результат, а результат правильный.

Демонстрация выполнения работы программного изделия (рисунки 27 – 30).

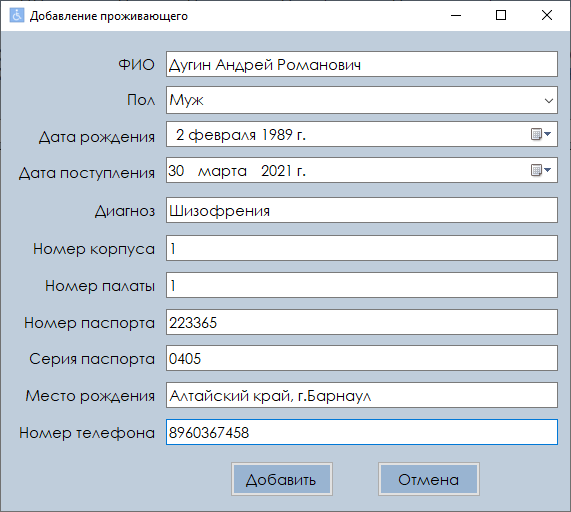


Рисунок 27 – Внесение данных в текстовые поля

После нажатия кнопки «Добавить», программное изделие подключается к базе данных, находит нужную таблицу и вносит данные (рисунок 28 - 29).

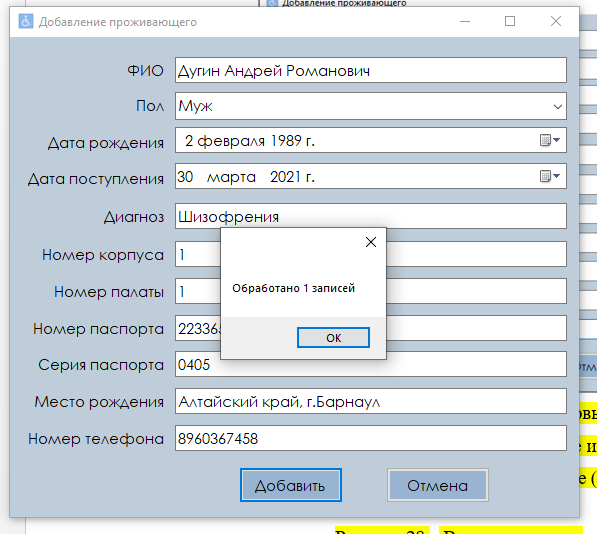


Рисунок 28 – Внесение данных

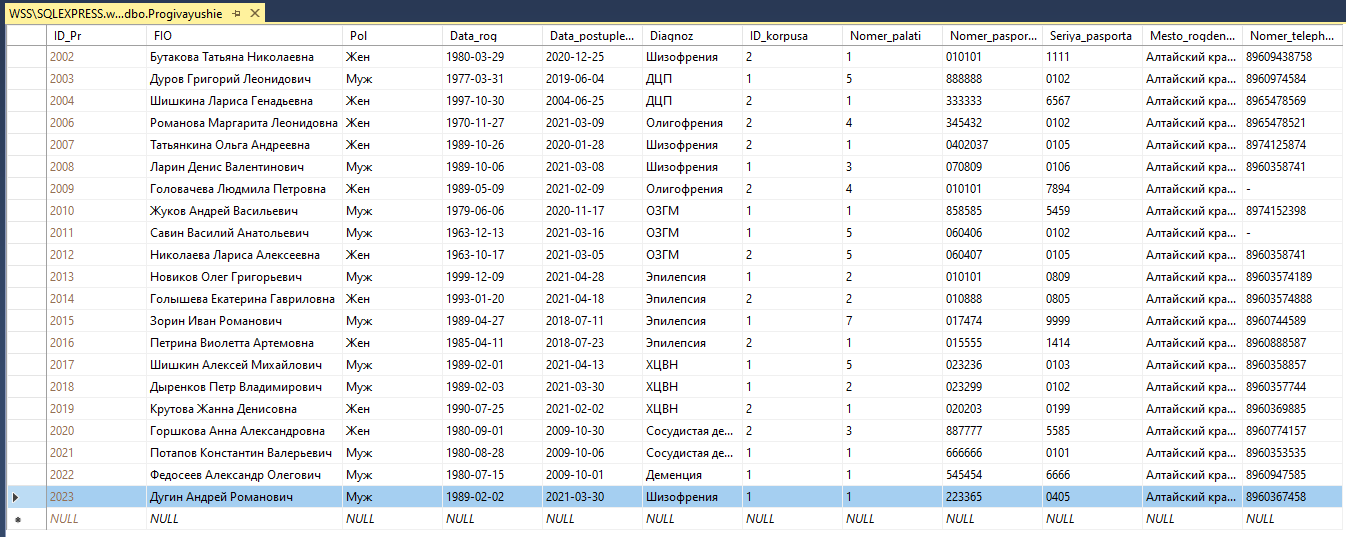


Рисунок 29 – Проверка внесения данных

При необходимости, через главную форму программы можно осуществить поиск информации.

Демонстрация работы поиска представлена на рисунке 30.

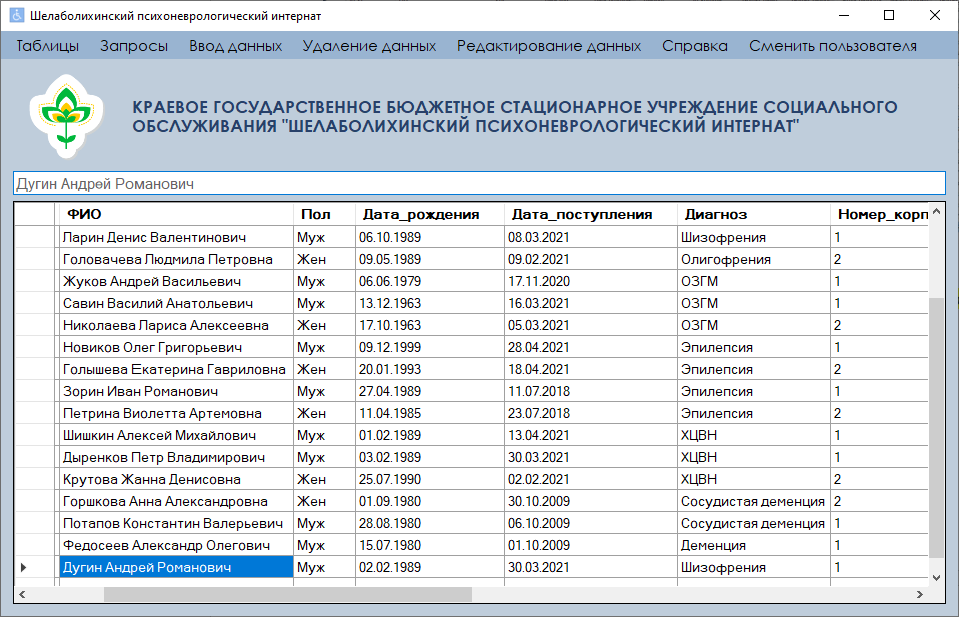


Рисунок 30 – Демонстрация работы поиска записи

**Раздел 4 Разработка проектных решений**

Программное изделие создано с использованием языка ООП высшего уровня C# в интегрированной среде разработки программного обеспечения Visual Studio 2017 Enterprise.

Преимущества языка C#:

* большое количество библиотек, которые уже имеются в базе;
* полная поддержка использования программных интерфейсов;
* управление памятью производится автоматически;
* наличие атрибутов;
* простой синтаксис и конструкции.

Программное изделие выполняет все поставленные на этапе разработки задачи.

Программное изделие функционирует на ОС Windows версии 7/8/8.1/10 с предустановленным .NETFramework v.4.6.1 и выше.

Системные требования минимальны, и описаны в руководстве программиста (Приложение У).

**Заключение**

В ходе создания дипломного проекта было разработано программное изделие для организации КГБСУСО «Шелаболихинский психоневрологический интернат».

Программное изделие в ходе работы выполняет следующие функции:

* создание, редактирование и удаление записей таблицы «Проживающие»;
* создание, редактирование и удаление записей таблицы «Данные о родственниках»;
* редактирование записей таблицы «Корпуса»;
* создание, редактирование и удаление записей таблицы «Медицинские работники»;
* поиск по записям таблицы «Проживающие»;
* поиск по записям таблицы «Данные о родственниках»;
* поиск по записям таблицы «Корпуса»;
* поиск по записям таблицы «Медицинские работники»;
* вывод таблицы «Проживающие» с фильтрацией по диагнозу.

Интерфейс программного изделия соответствует всем требованиям заказчика и по максимуму упрощён, для беспроблемной работы пользователя.

Программное изделие принято на использование в организации. Все используемые в программном изделии модули выполняют свою функцию и поставленную задачу, в ходе вычислений выдают верный результат, из чего можно заключить, что цель дипломного проекта выполнена.

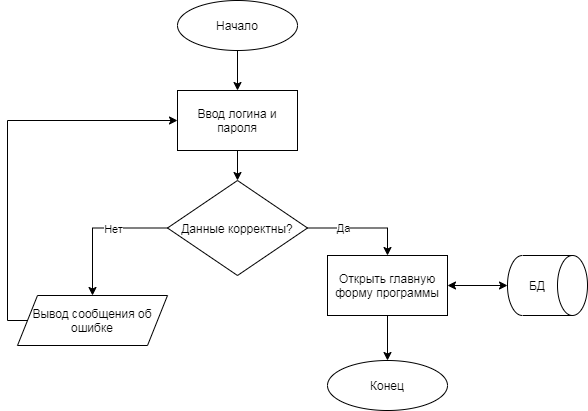
По желанию заказчика программное изделие будет дорабатываться, при выявлении ошибок они будут в кратчайшие сроки устраняться.

**Список использованных источников**

1. ГОСТ 19.001-77.ЕСПД. Общие положения. -Введен 1980-01-01.-Москва: Стандартинформ,2010.
2. ГОСТ 19.401-78.ЕСПД. Текст программы. Требования к содержанию и оформлению pdf. -Введен 1989-01-01. - Москва: Стандартинформ,2010.
3. ГОСТ 19.701-90.ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения.-Введен 1992-01-01.- Москва: Стандартинформ,2010.
4. ГОСТ 19.202-78.ЕСПД.Техническое задание. Требования к содержанию и оформлению.- Введен 1980-01-01.-Москва: Стандартинформ,2010.
5. ГОСТ 19.504-79.ЕСПД. Руководство программиста. Требования к содержанию и оформлению.- Введен 1980-01-01.-Москва: Стандартинформ, 2010.
6. Гниденко И.Г. Технология разработки программного обеспечения: учеб. Пособие для спо/ И.Г.Гниденко. -Москва:Юрайт,2017.-235с.
7. Рудаков, А. В. Технология разработки программных продуктов : учебник для спо / А. В. Рудаков. - 10-е изд., перераб. и доп. - Москва : Академия, 2016. - 208 с.
8. Рудаков, А. В. Технология разработки программных продуктов : практикум / А. В. Рудаков, Г.Н.Федорова. - Москва : Академия, 2014. - 192 с.
9. Федорова Г.Н. Участие в интеграции программных модулей :учеб. пособие для спо /Г.Н.Федорова.-Москва:Академия,2016.-304с.
10. Федорова Г.Н. Разработка программных модулей программного обеспечения для компьютерных систем: учебник для спо.-Москва:Академия,2019.-336с.
11. Рудаков, А. В. Технология разработки программных продуктов : учебник для спо / А. В. Рудаков. - 10-е изд., перераб. и доп. - Москва : Академия, 2019. - 208 с.
12. MSDN-сеть разработчиков Microsoft: Каталог API (Microsoft) и справочных материалов [Электронный ресурс].-Электрон.дан.-Режим доступа:https:// microsoft.com/ library.- Загл.с экрана.
13. Форум программистов и сисадмистов Киберфорум [Электронный ресурс].- Электрон.дан.-Режим доступа:https:// www/cyberforum.ru.- Загл.с экрана.
14. Единая система программной документации: Стандарты ЕСПД [Электронный ресурс].- Электрон.дан.-Режим доступа:https://www/swrit.ru/gost-espd.html.- Загл.с экрана.

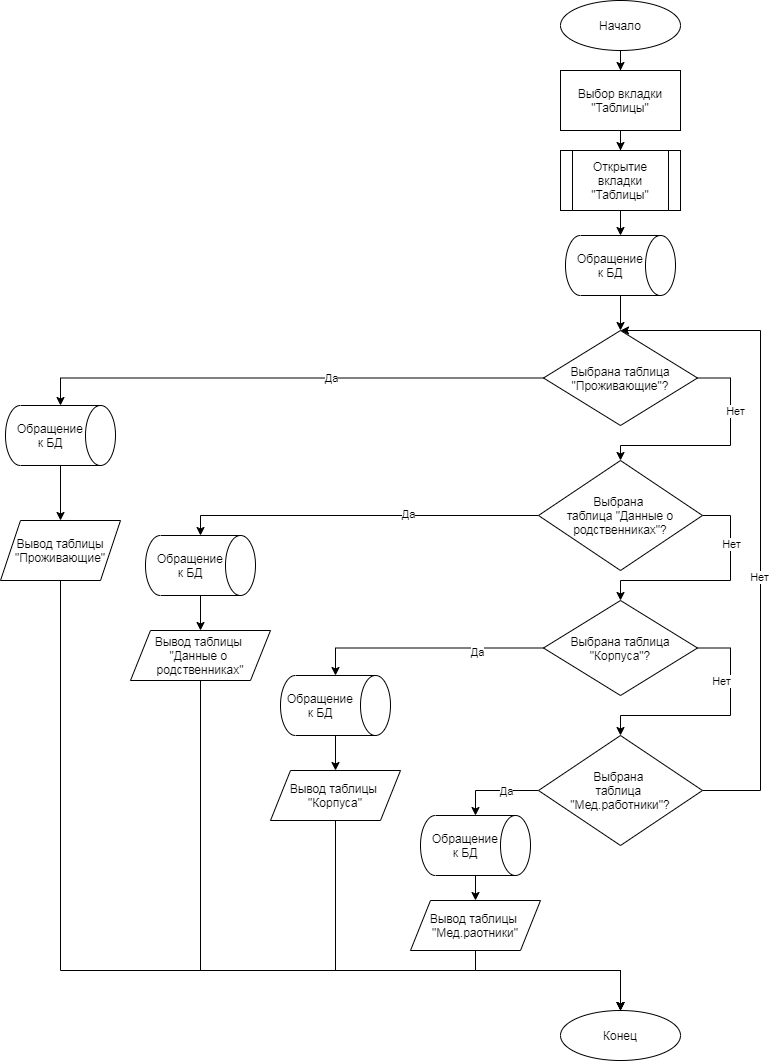
**Приложение А**

# Алгоритм формы авторизации



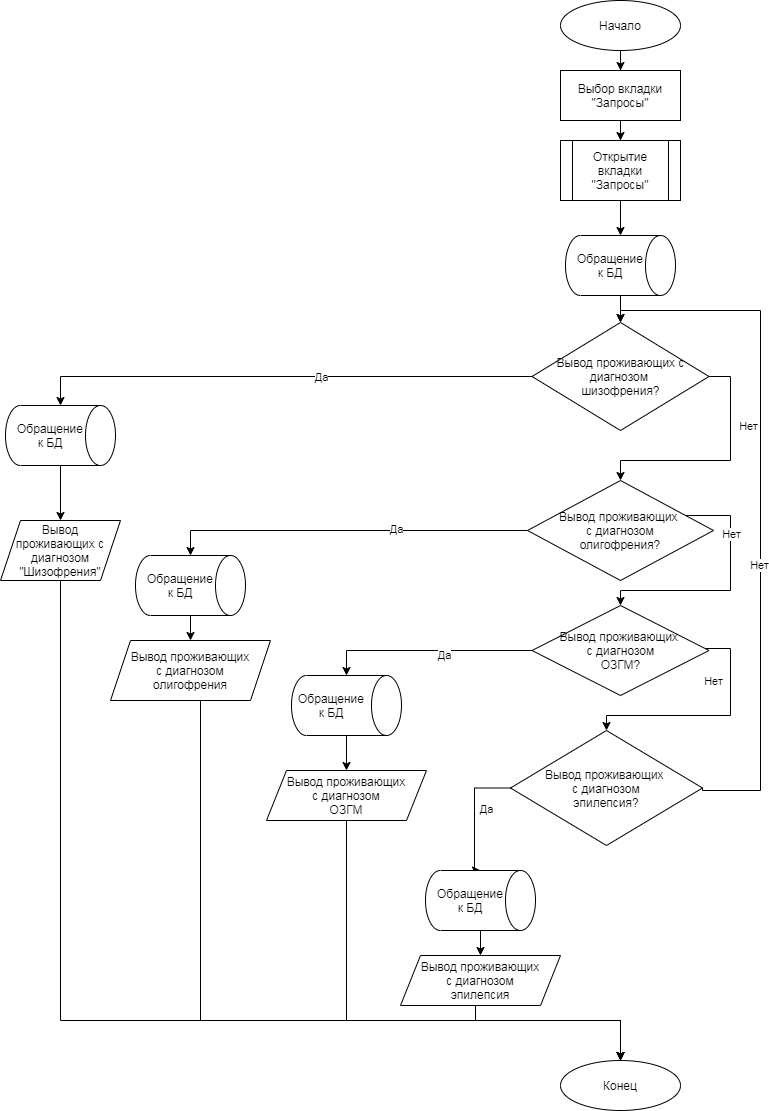
**Приложение Б**

# Алгоритм работы вывода таблиц

****

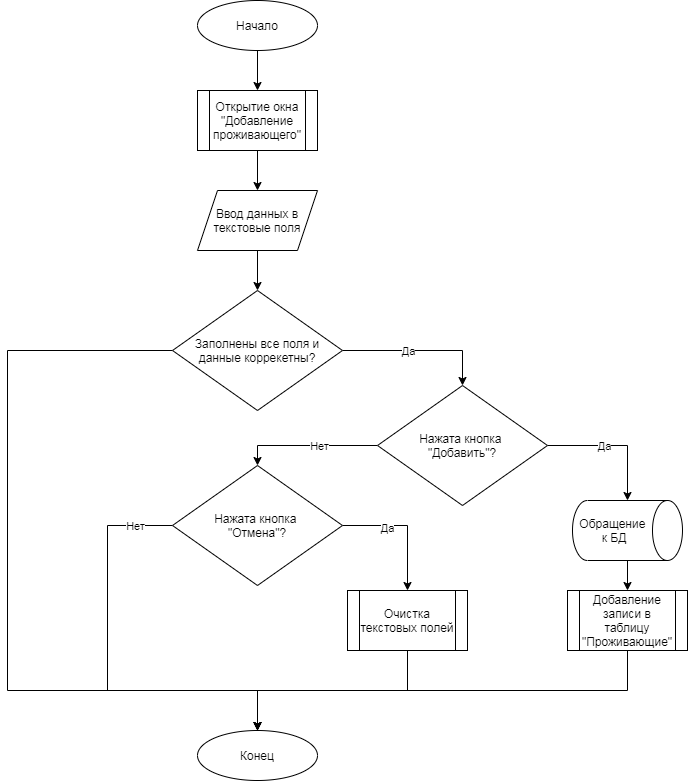
**Приложение В**

# Алгоритм работы вкладки «Запросы»

****

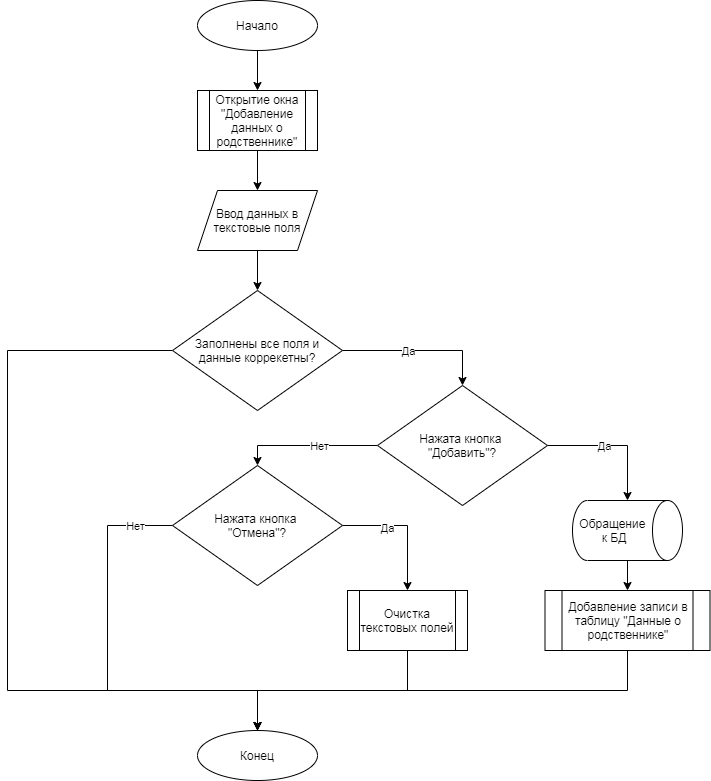
**Приложение Г**

**Алгоритм добавления записи проживающего**

****

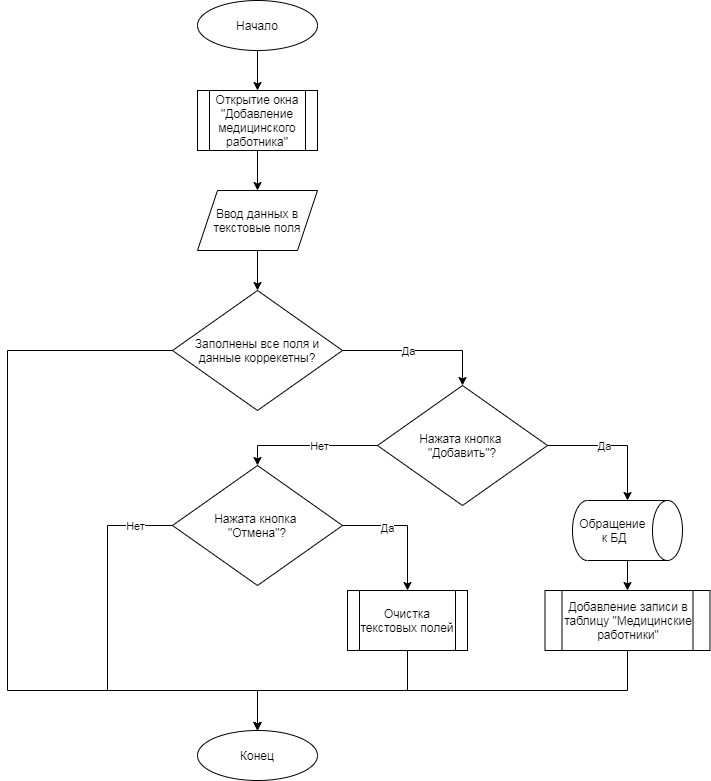
**Приложение Д**

**Алгоритм добавления записи данных о родственнике**



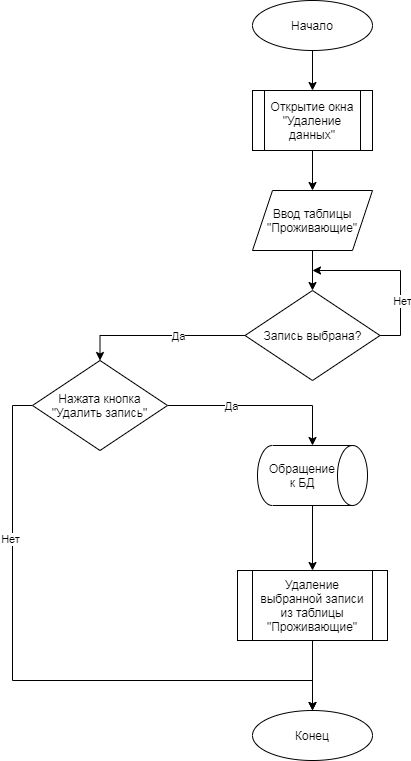
**Приложение Е**

**Алгоритм добавления записи медицинского работника**

****

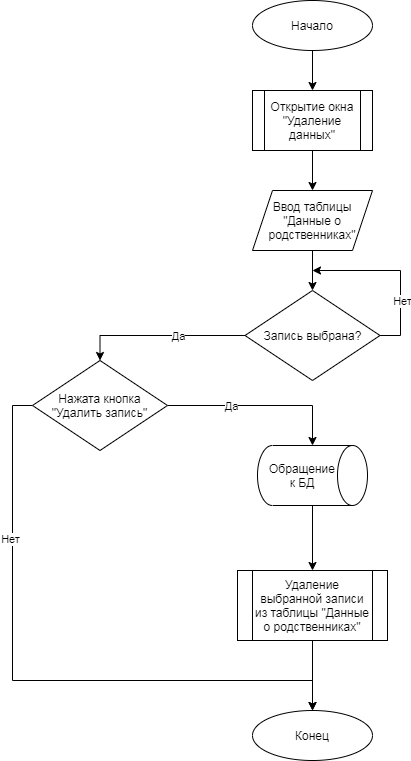
**Приложение Ж**

**Алгоритм работы удаления записи проживающего**



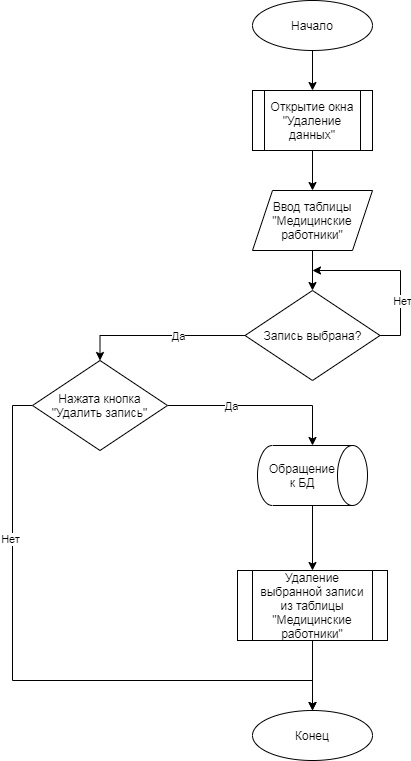
# Приложение И

# Алгоритм работы удаления данных о родственнике

****

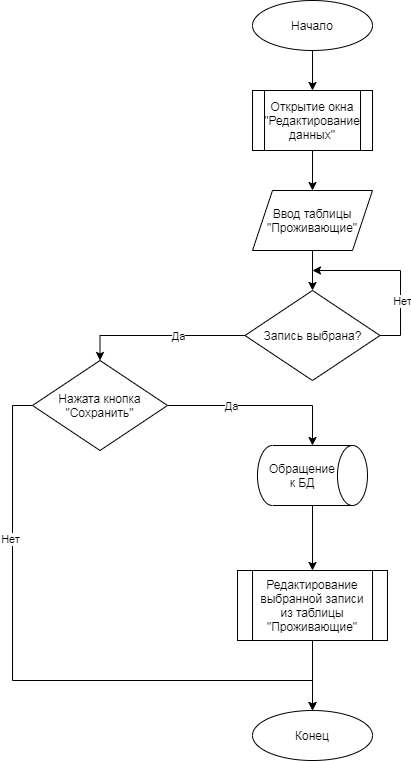
**Приложение К**

**Алгоритм работы удаления записи медицинского работника**

****

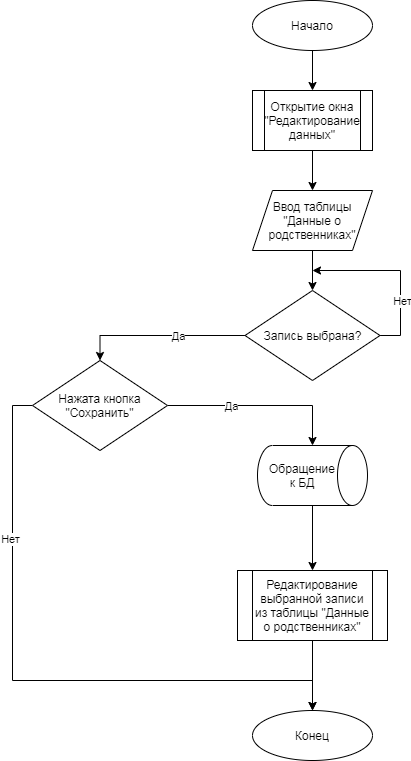
**Приложение Л**

**Алгоритм работы редактирования записи проживающего**



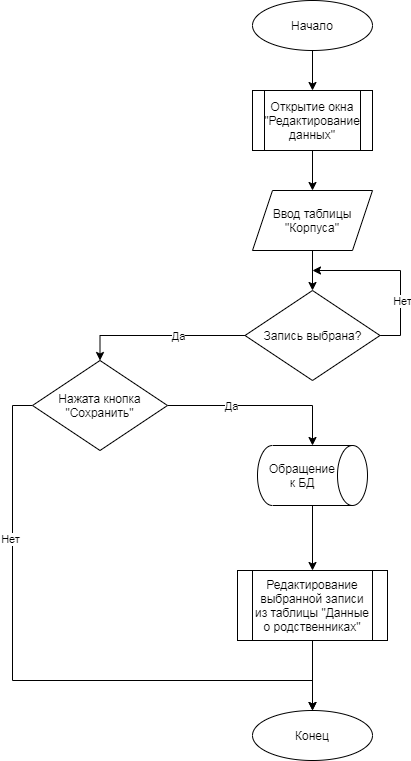
**Приложение М**

**Алгоритм работы редактирования данных о родственнике**

****

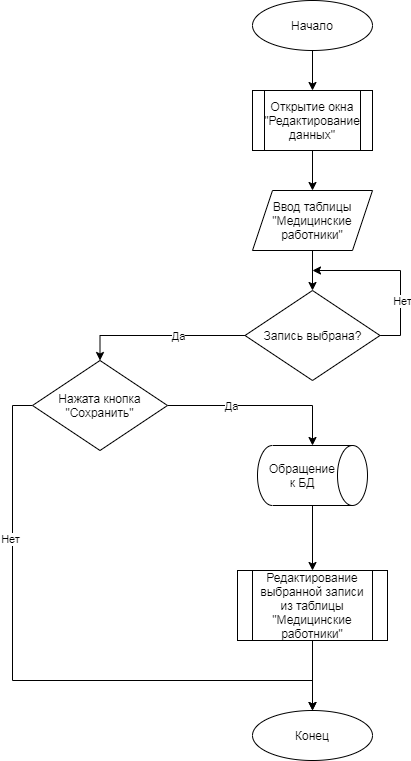
**Приложение Н**

**Алгоритм работы редактирования записи корпуса**



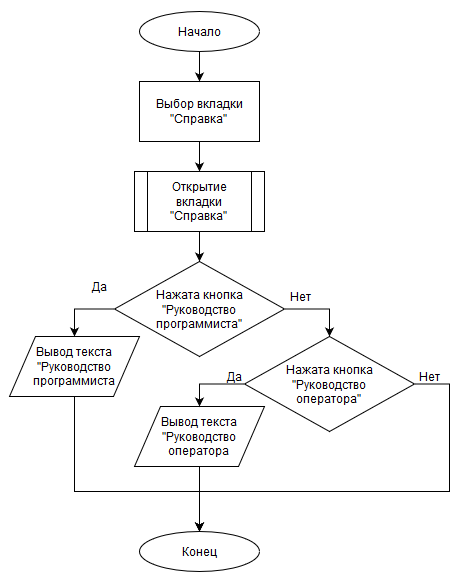
# Приложение П

# Алгоритм работы редактирования записи медицинского работника

****

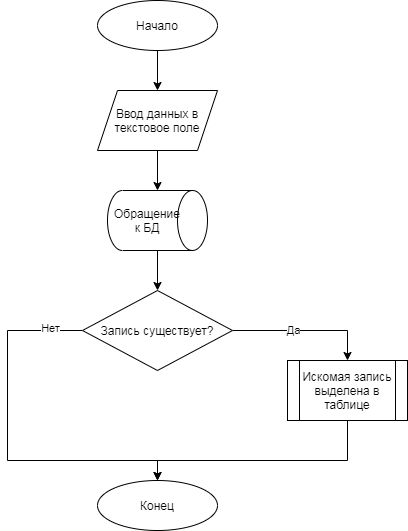
**Приложение Р**

**Алгоритм работы окна «Справка»**



**Приложение С**

**Алгоритм работы поиска записи**

****

**Приложение Т**

**Руководство оператора**

Руководство пользователя по применению и эксплуатации программы для организации КГБСУСО «Шелаболихинский психоневрологический интернат», предназначенной для ведения учёта проживающих, их близких родственников, а так же медицинских работников и корпусов интерната.

В данном программном документе, в разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы (минимальный состав аппаратных и программных средств и т.п.).

В данном программном документе, в разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды.

В разделе «Сообщения оператору» приведены тексты сообщений, выдаваемых в ходе выполнения программы, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

**1. Назначение программы**

**1.1 Функциональное назначение программы**

Основной функцией программы является ведение учёта проживающих, их близких родственников, а так же медицинских работников и корпусов интерната. Имеется возможность добавления, удаления, редактирования данных, а также осуществлять поиск по записям.

**1.2 Эксплуатационное назначение программы**

Основное назначение программы – добавление информации в документы, которые необходимы в процессе деятельности организации.

**1.3** **Состав функций**

**1.3.1 Авторизация**

Авторизация – представление пользователя системе управления программы.

Для того чтобы авторизироваться в программе пользователь должен ввести в специальные поля логин и пароль и нажать на кнопку «Войти».

Окно авторизации представлено на рисунке Т.1.

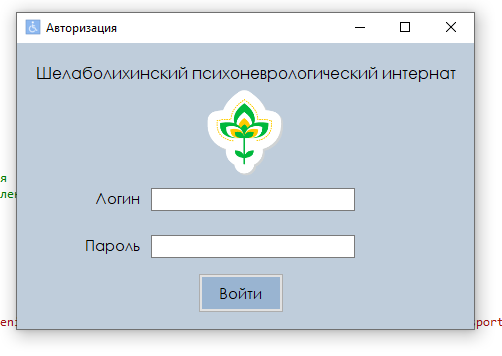


Рисунок Т.1 – Окно авторизации программы

1.3.2 Функция вывода таблиц

Осуществляет вывод информации из таблиц базы данных при нажатии на кнопку «Таблицы» из меню и далее на название нужной таблицы.

Главное окно программы представлено на рисунке Т.2.

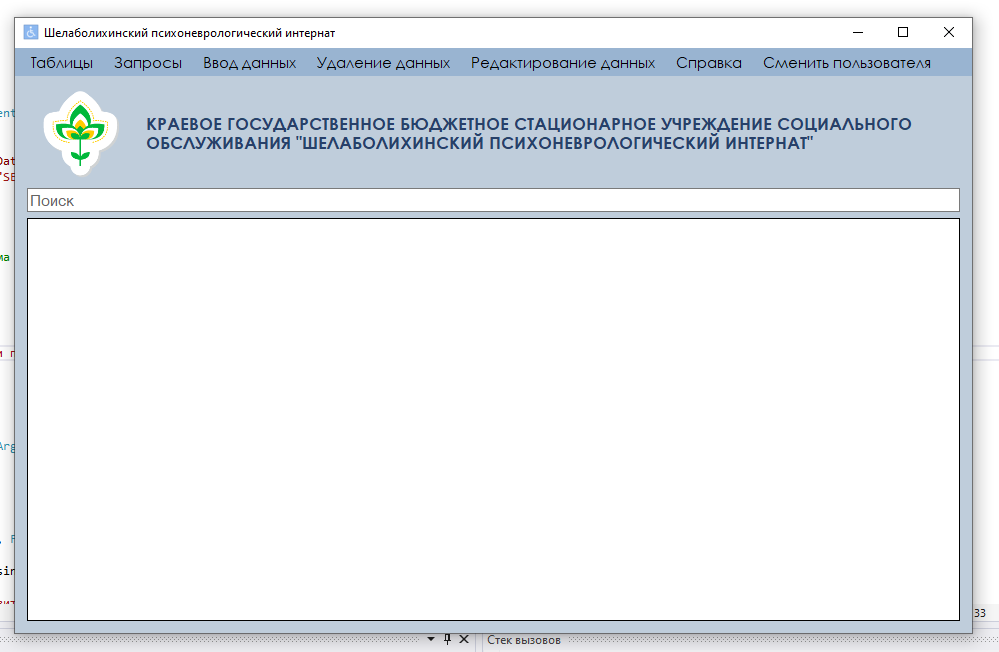


Рисунок Т.2 – Главное окно программы

Главное окно программы с открытой вкладкой «Таблицы» представлено на рисунке Т.3.

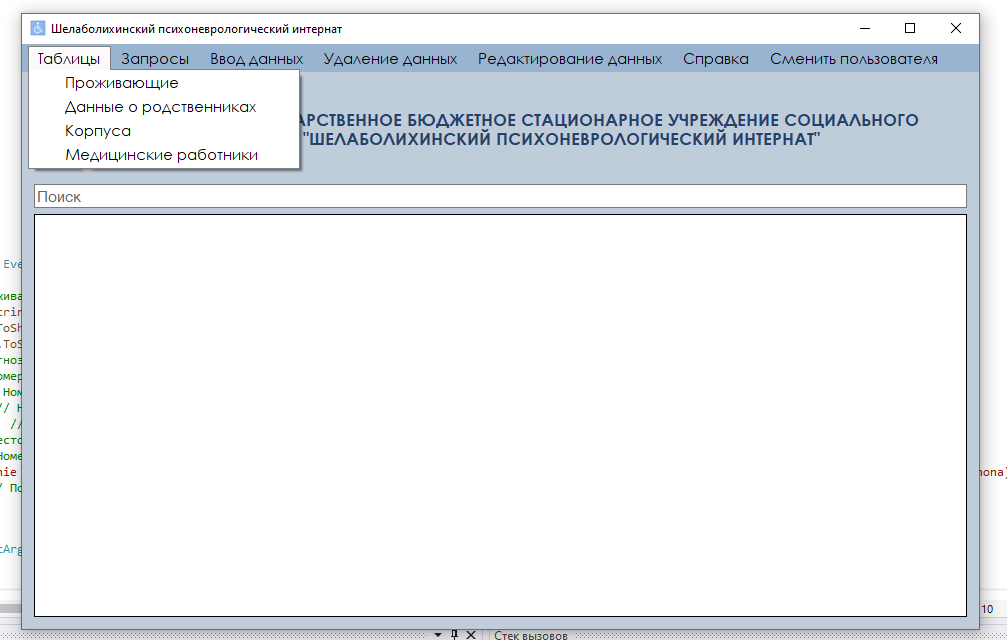


Рисунок Т.3 – Главное окно программы с открытой вкладкой «Таблицы»

Главное окно программы с выведенной таблицей «Проживающие» представлено на рисунке Т.4.

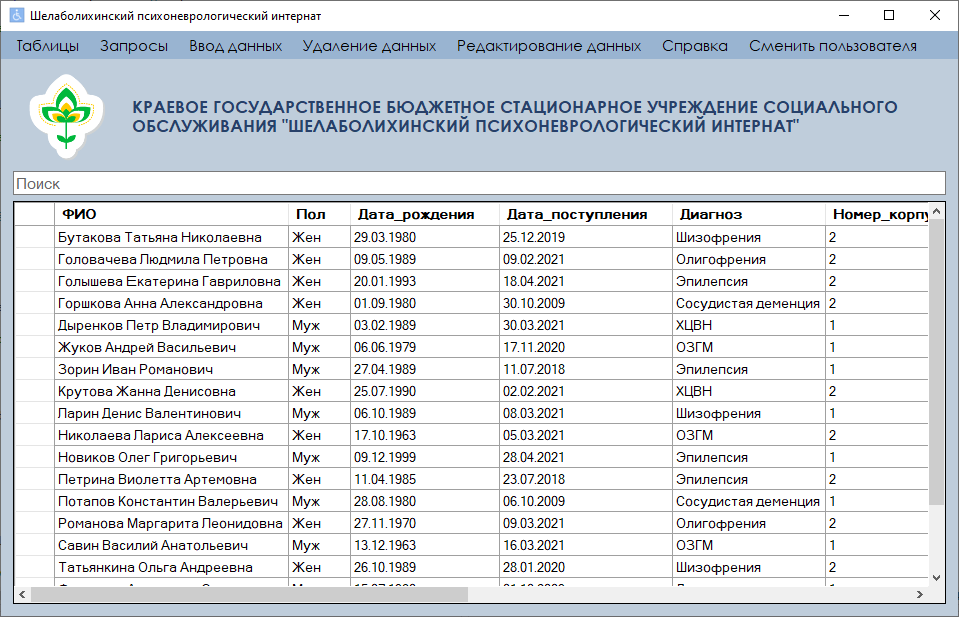


Рисунок Т.4 – Главное окно программы с выведенной таблицей «Проживающие»

Главное окно программы с выведенной таблицей «Данные о родственниках» представлено на рисунке Т.5.

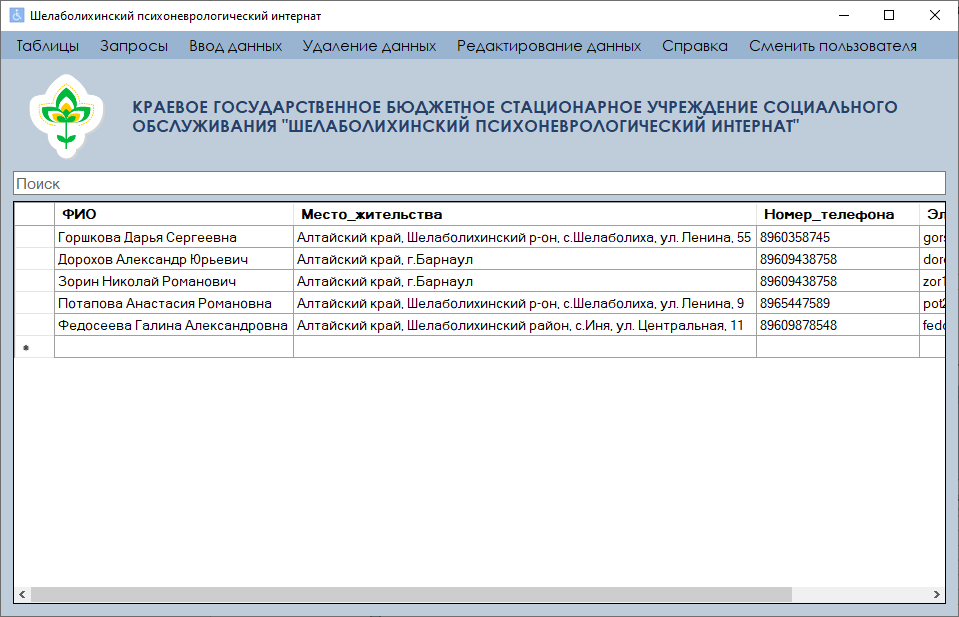


Рисунок Т.5 – Главное окно программы с выведенной таблицей «Данные о родственниках»

Главное окно программы с выведенной таблицей «Корпуса» представлено на рисунке Т.6.

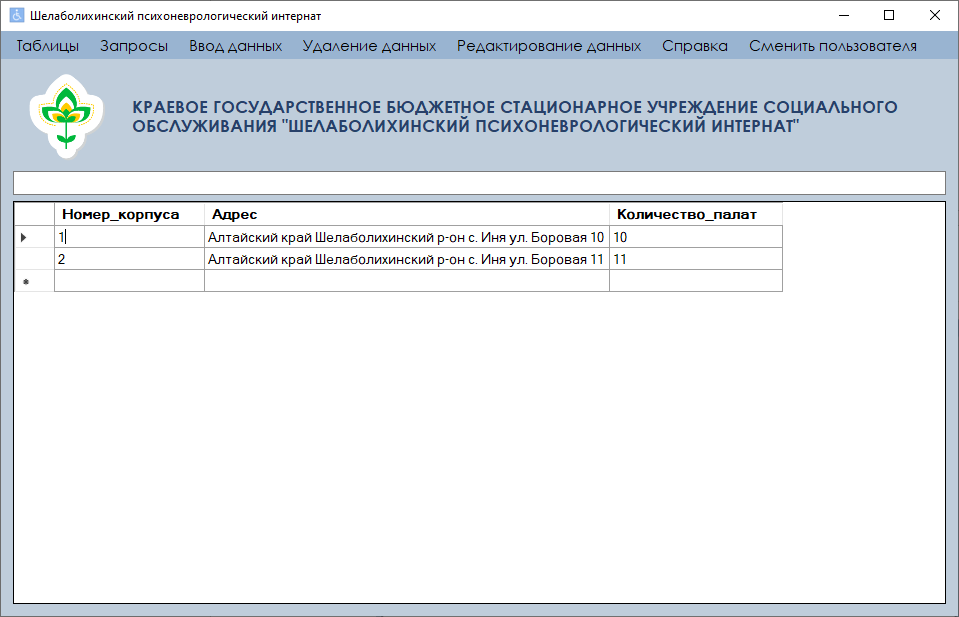


Рисунок Т.6 – Главное окно программы с выведенной таблицей «Корпуса»

Главное окно программы с выведенной таблицей «Медицинские работники» представлено на рисунке Т.7.

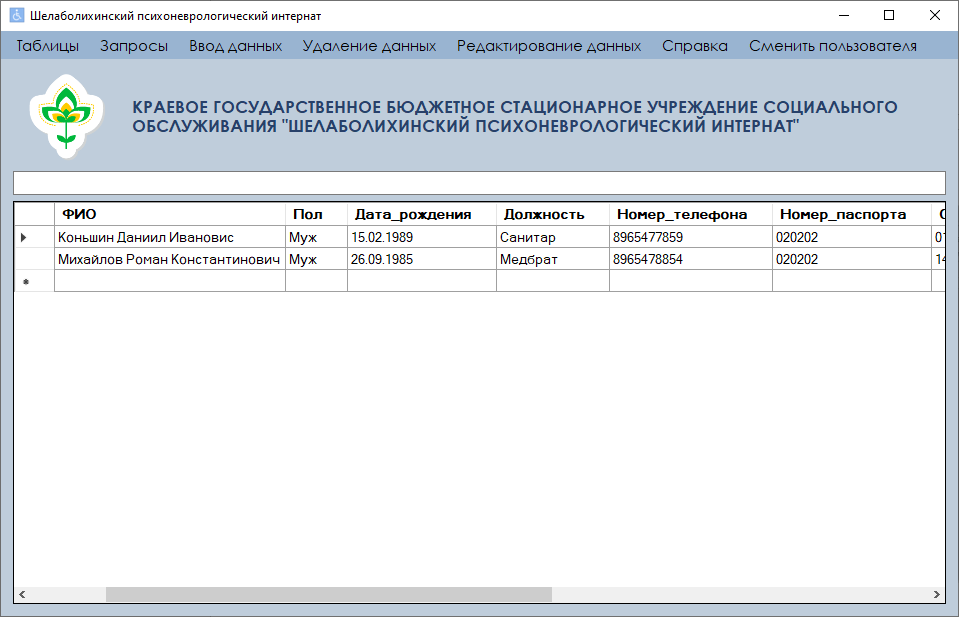


Рисунок Т.7 – Главное окно программы с выведенной таблицей «Медицинские работники»

**1.3.3 Функция вывода таблицы «Проживающие» с фильтрацией по диагнозу**

Осуществляет вывод проживающих с фильтрацией по определенному диагнозу по нажатию на вкладку «Запросы» и далее на кнопку для вывода проживающих с нужным диагнозом.

Главное окно программы с открытым меню вкладки «Запросы» представлено на рисунке Т.8.

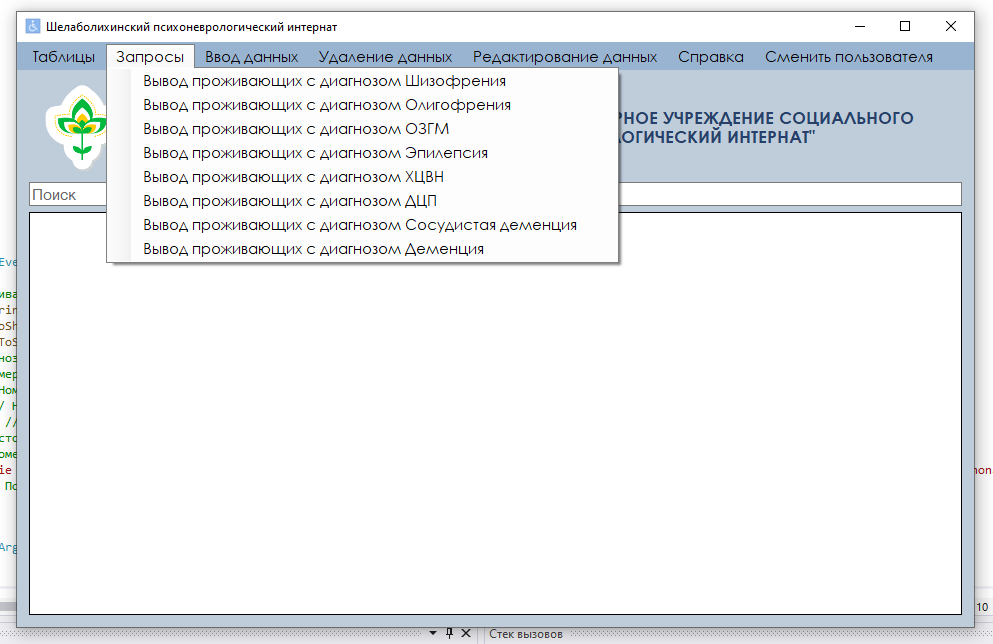


Рисунок Т.8 – Главное окно программы с открытым меню вкладки «Запросы»

Пример результата выполнения запроса представлен на рисунке Т.9.

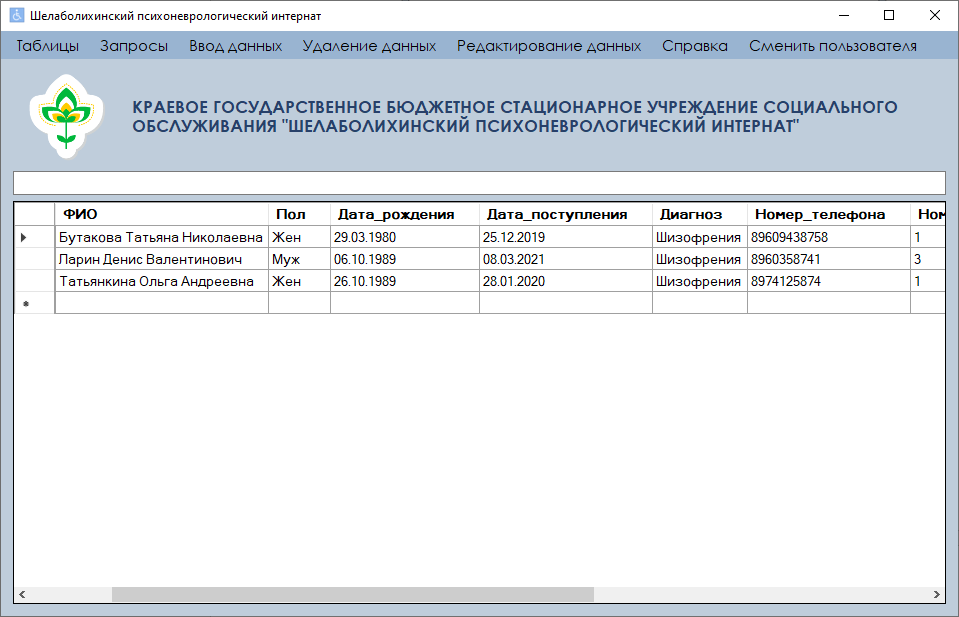


Рисунок Т.9 – Вывод проживающих с диагнозом шизофрения

**1.3.4 Функция добавление записи проживающего**

Осуществляет добавление записи в таблицу «Проживающие» базы данных при переходе на вкладку «Ввод данных» > «Проживающие» по нажатию на кнопку «Добавить» после заполнения всех текстовых полей.

Окно добавления проживающего представлено на рисунке Т.10.

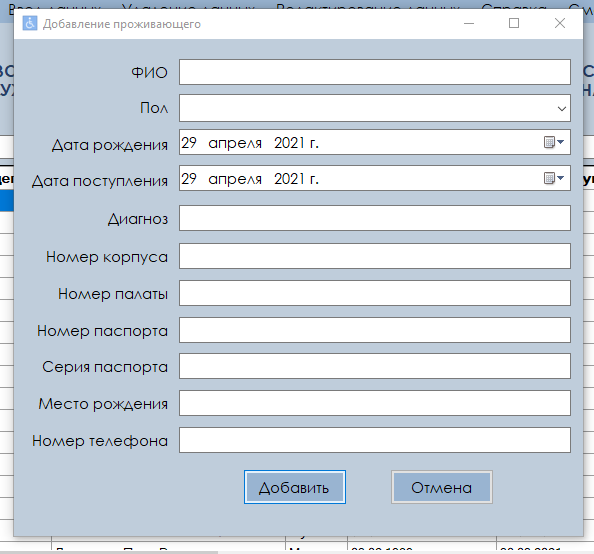


Рисунок Т.10 – Окно добавления проживающего

**1.3.5 Функция добавление родственника проживающего**

Осуществляет добавление записи в таблицу «Данные о родственниках» базы данных при переходе на вкладку «Ввод данных» > «Данные о родственниках» по нажатию на кнопку «Добавить» после заполнения всех текстовых полей.

Окно добавления родственника проживающего представлено на рисунке Т.11.

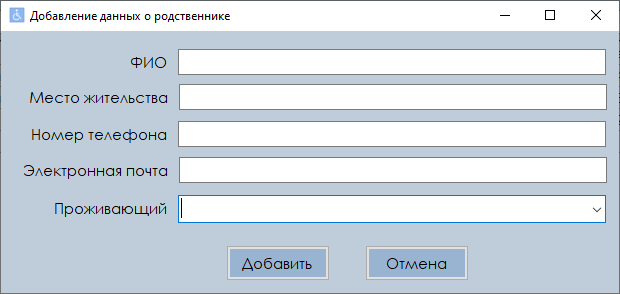


Рисунок Т.11 – Окно добавления родственника проживающего

**1.3.6 Функция добавление записи медицинского работника**

Осуществляет добавление записи в таблицу «Медицинские работники» базы данных при переходе на вкладку «Ввод данных» > «Медицинские работники» по нажатию на кнопку «Добавить» после заполнения всех текстовых полей.

Окно добавления медицинского работника представлено на рисунке Т.12.

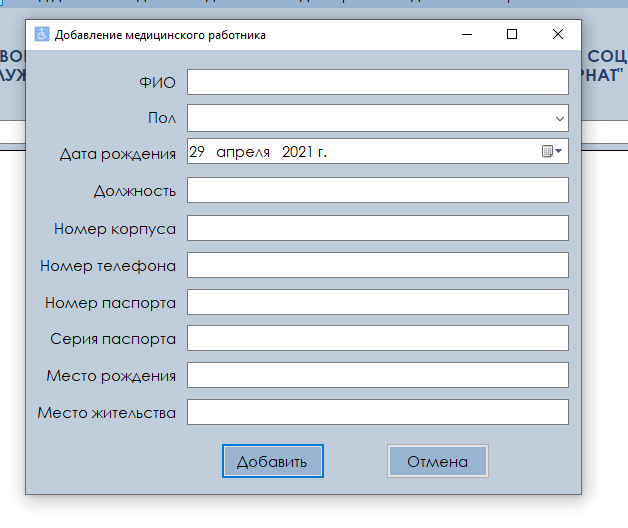


Рисунок Т.12 – Окно добавления медицинского работника

**1.3.7 Функция отмены добавления записи**

Осуществляет очистку полей формы по нажатию на кнопку «Отмена».

Кнопка «Отмена» представлена на рисунке Т.13.

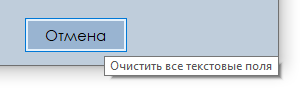


Рисунок Т.13 – Кнопка «Отмена»

**1.3.8 Функция удаления записи проживающего**

Осуществляет удаление выбранной записи из таблицы «Проживающие» на вкладке «Удаление данных» > «Проживающие» по нажатию на кнопку «Удалить».

Окно удаления записи проживающего с выбранной записью представлено на рисунке Т.14.

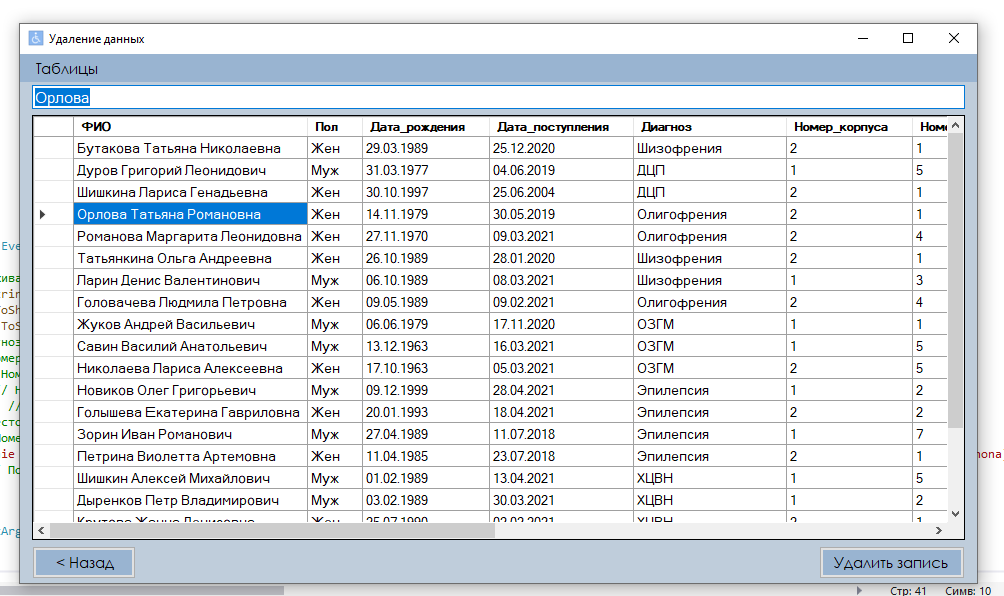


Рисунок Т.14 – Окно удаления записи проживающего с выбранной записью

Результат выполнения удаления записи проживающего представлен на рисунке Т.15.

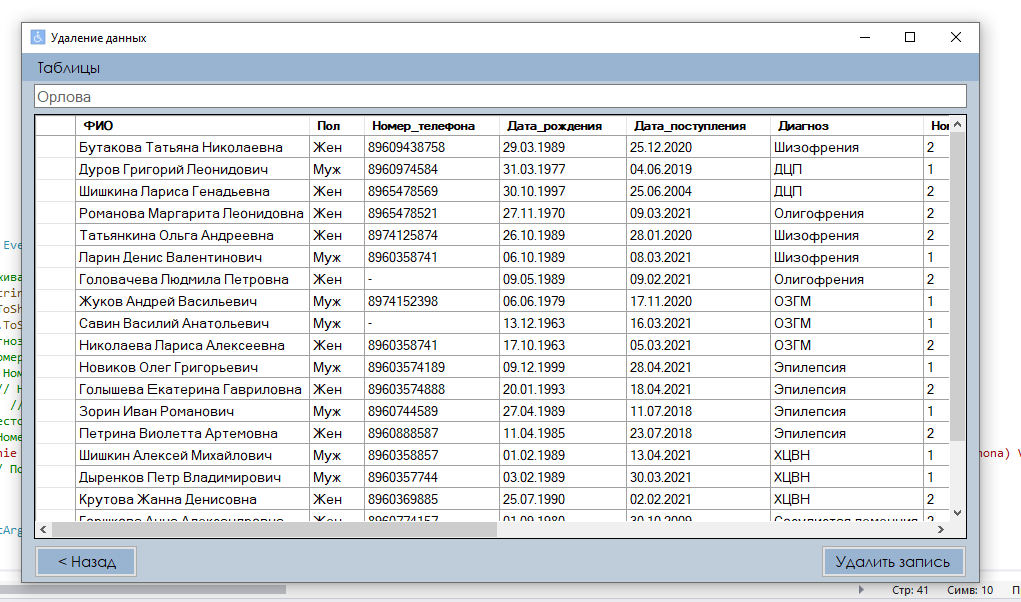


Рисунок Т.15 – Результат выполнения удаления записи проживающего

Кнопка удаления записи проживающего представлена на рисунке Т.16.

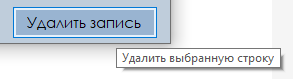


Рисунок Т.16 – Кнопка удаления

**1.3.9 Функция удаления записи родственника проживающего**

Осуществляет удаление выбранной записи из таблицы «Данные о родственниках» на вкладке «Удаление данных» > «Данные о родственниках» по нажатию на кнопку «Удалить».

Окно удаления записи родственника проживающего с выбранной записью представлено на рисунке Т.17.

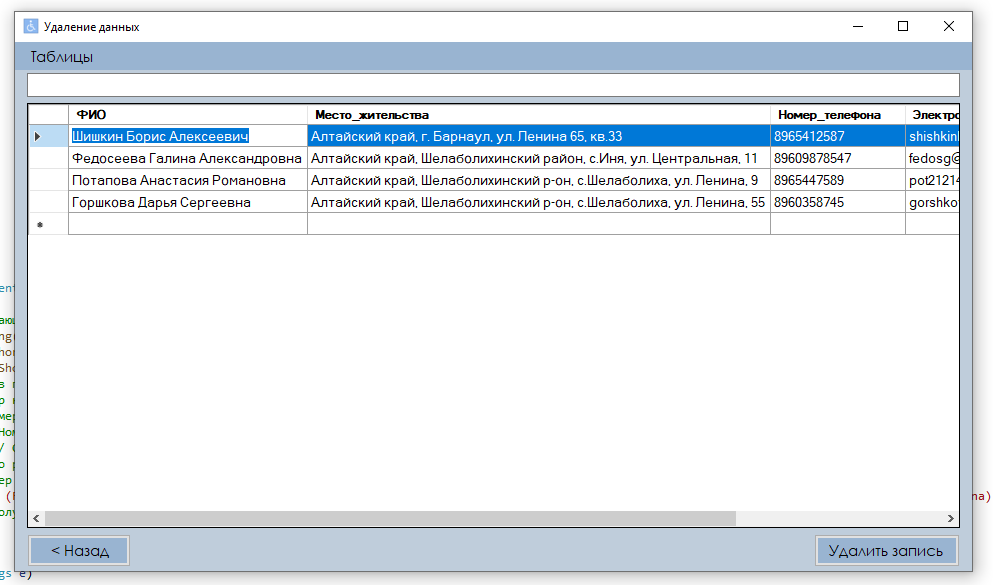


Рисунок Т.17 – Окно удаления записи родственника проживающего с выбранной записью

Результат выполнения удаления записи родственника проживающего представлен на рисунке Т.18.

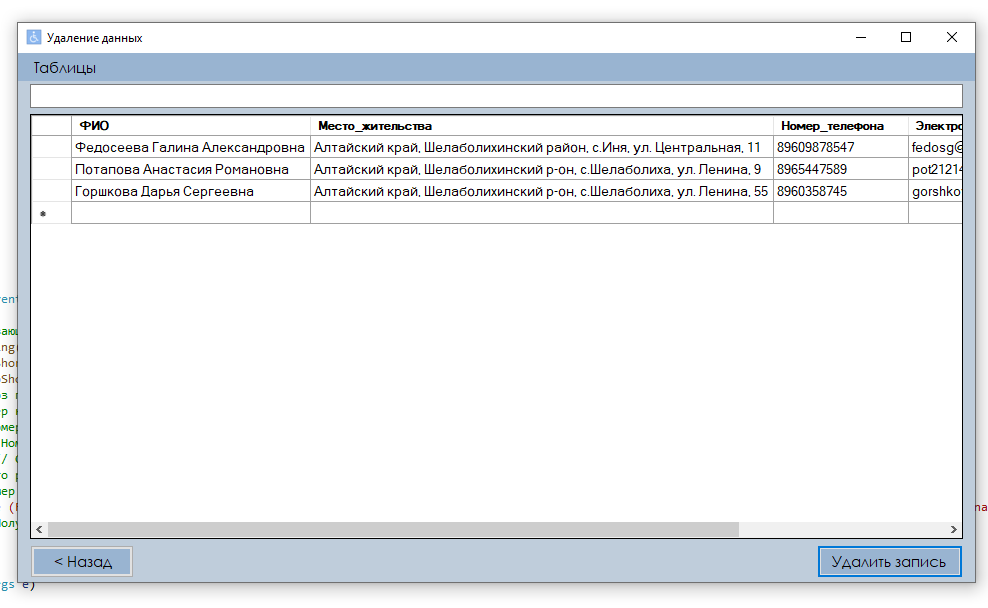


Рисунок Т.18 – Результат выполнения удаления записи родственника проживающего

Кнопка удаления записи родственника проживающего представлена на рисунке Т.19.

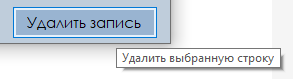


Рисунок Т.19 – Кнопка удаления

**1.3.10 Функция удаления записи медицинского работника**

Осуществляет удаление выбранной записи из таблицы «Медицинские работники» на вкладке «Удаление данных» > «Медицинские работники» по нажатию на кнопку «Удалить».

Окно удаления записи медицинского работника с выбранной записью представлено на рисунке Т.20.

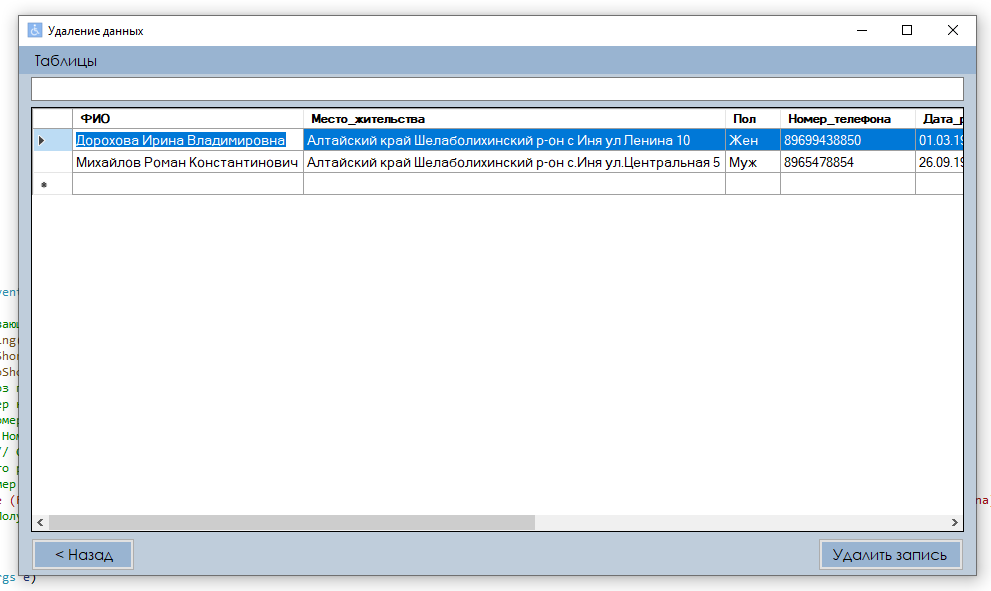


Рисунок Т.20 – Окно удаления записи медицинского работника с выбранной записью

Результат выполнения удаления записи медицинского работника представлен на рисунке Т.21.

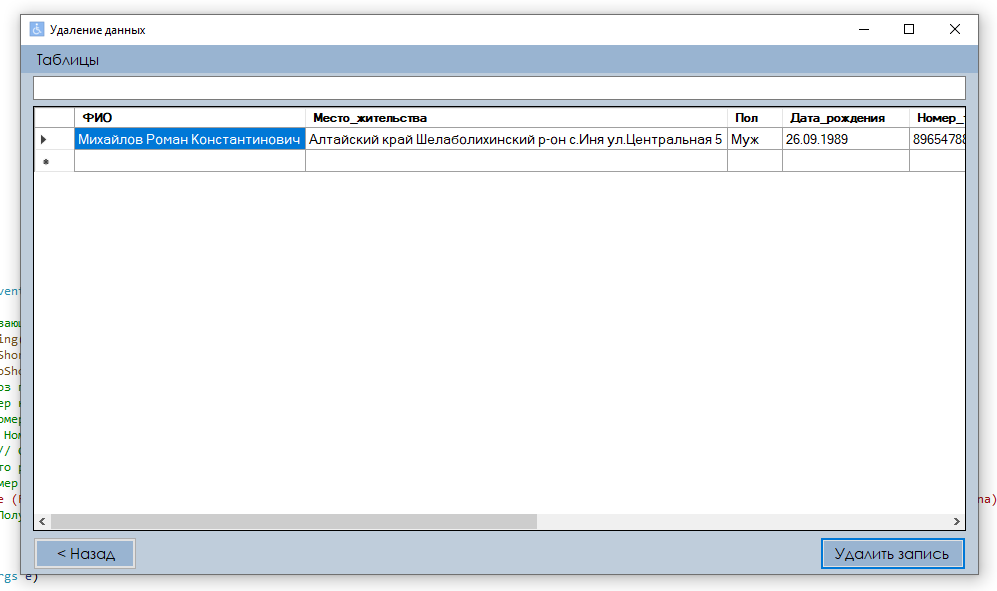


Рисунок Т.21 – Результат выполнения удаления записи медицинского работника

Кнопка удаления записи медицинского работника представлена на рисунке Т.22.

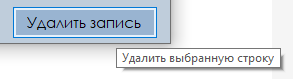


Рисунок Т.22 – Кнопка удаления

**1.3.11 Функция редактирования записи проживающего**

Осуществляет изменение выбранной записи на вкладке «Редактирование данных» в таблице «Проживающие» базы данных по нажатию на кнопку «Сохранить». Таким образом программный продукт перезаписывает ранее находящуюся в записи информацию.

Окно редактирование данных с выведенной таблицей «Проживающие» представлено на рисунке Т.23.

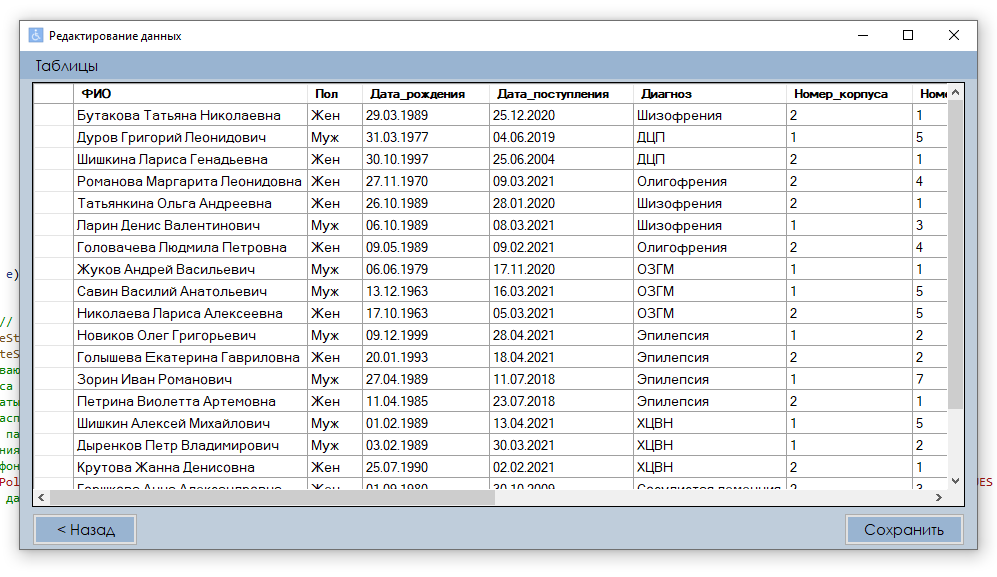


Рисунок Т.23 – Окно редактирование данных с выведенной таблицей «Проживающие»

Выбранная ячейка для редактирования данных таблицы «Проживающие» представлена рисунке Т.24.

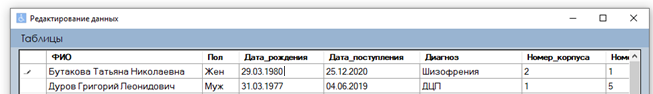


Рисунок Т.24 – Выбранная ячейка для редактирования данных таблицы «Проживающие»

Выбранная ячейка таблицы «Проживающие» с внесенными изменениями представлена на рисунке Т.25.

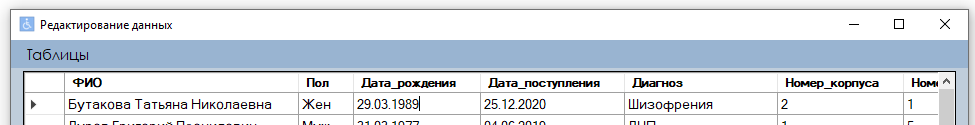


Рисунок Т.25 – Выбранная ячейка таблицы «Проживающие» с внесенными изменениями

Кнопка для сохранения внесенных изменений представлена на рисунке Т.26.

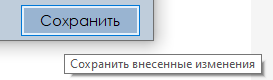


Рисунок Т.26 – Кнопка для сохранения внесенных изменений

**1.3.12 Функция редактирования записи родственника проживающего**

Осуществляет изменение выбранной записи на вкладке «Редактирование данных» в таблице «Данные о родственниках» базы данных по нажатию на кнопку «Сохранить». Таким образом программный продукт перезаписывает ранее находящуюся в записи информацию.

Окно редактирование данных с выведенной таблицей «Данные о родственниках» представлено на рисунке Т.27.

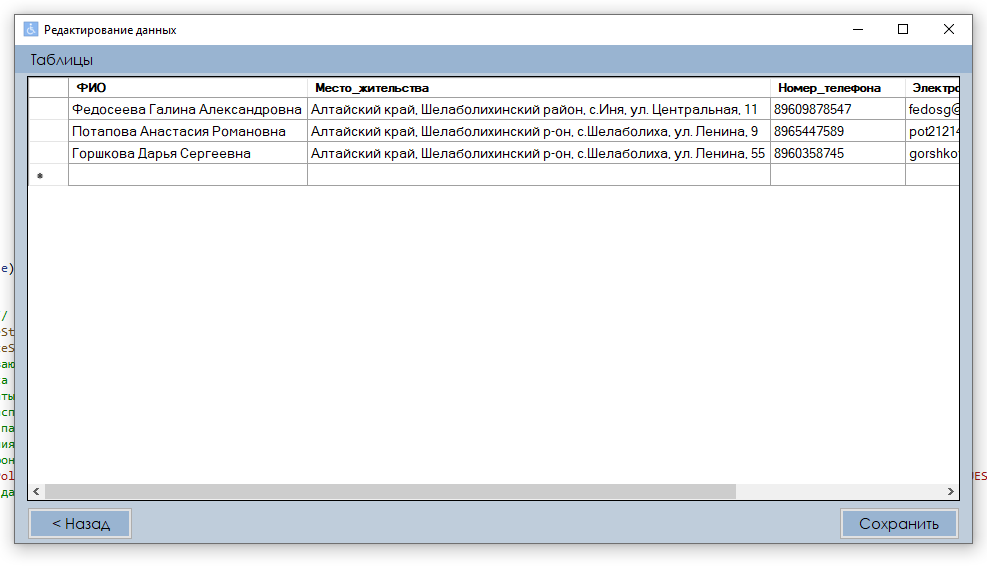


Рисунок Т.27 – Окно редактирование данных с выведенной таблицей «Данные о родственниках»

Выбранная ячейка для редактирования данных таблицы «данные о родственниках» представлена рисунке Т.28.

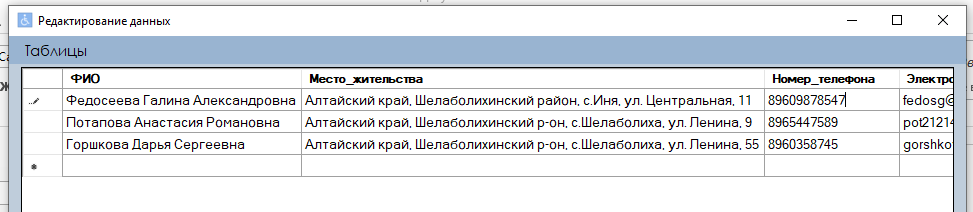


Рисунок Т.28 – Выбранная ячейка для редактирования данных таблицы «Данные о родственниках»

Выбранная ячейка таблицы «Данные о родственниках» с внесенными изменениями представлена на рисунке Т.29.

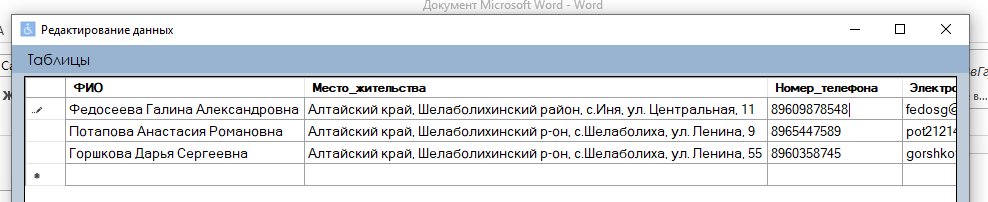


Рисунок Т.29 – Выбранная ячейка таблицы «Данные о родственниках» с внесенными изменениями

Кнопка для сохранения внесенных изменений представлена на рисунке Т.30.

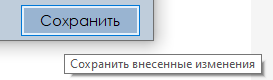


Рисунок Т.30 – Кнопка для сохранения внесенных изменений

**1.3.13 Функция редактирования записи корпуса**

Осуществляет изменение выбранной записи на вкладке «Редактирование данных» в таблице «Корпуса» базы данных по нажатию на кнопку «Сохранить». Таким образом, программный продукт перезаписывает ранее находящуюся в записи информацию.

Окно редактирование данных с выведенной таблицей «Корпуса» представлено на рисунке Т.31.

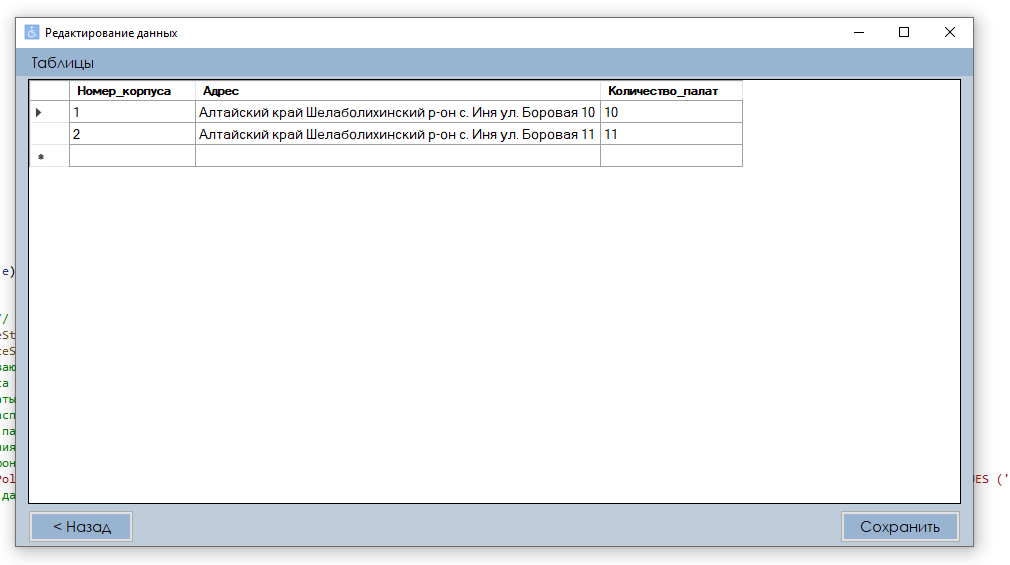


Рисунок Т.31 – Окно редактирование данных с выведенной таблицей «Корпуса»

Выбранная ячейка для редактирования данных таблицы «Корпуса» представлена рисунке Т.32.

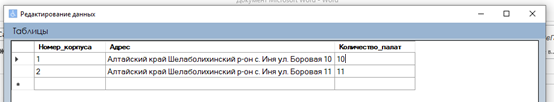


Рисунок Т.32 – Выбранная ячейка для редактирования данных таблицы «Корпуса»

Выбранная ячейка таблицы «Корпуса» с внесенными изменениями представлена на рисунке Т.33.

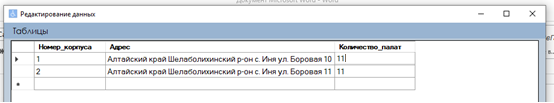


Рисунок Т.33 – Выбранная ячейка таблицы «Корпуса» с внесенными изменениями

Кнопка для сохранения внесенных изменений представлена на рисунке Т.34.

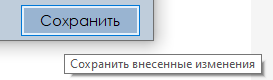


Рисунок Т.34 – Кнопка для сохранения внесенных изменений

**1.3.14 Функция редактирования записи медицинского работника**

Осуществляет изменение выбранной записи на вкладке «Редактирование данных» в таблице «Медицинские работники» базы данных по нажатию на кнопку «Сохранить». Таким образом, программный продукт перезаписывает ранее находящуюся в записи информацию.

Окно редактирование данных с выведенной таблицей «Медицинские работники» представлено на рисунке Т.35.

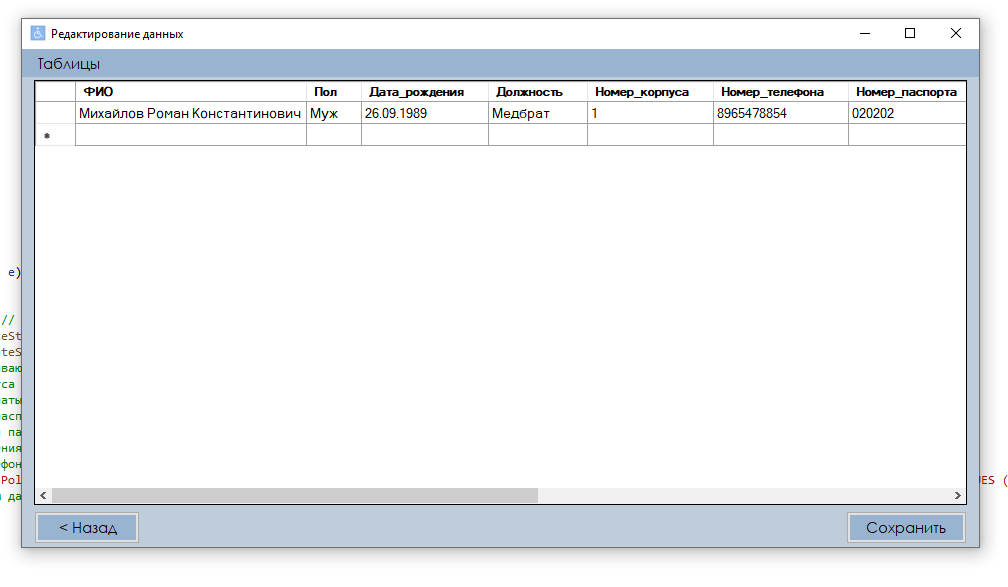


Рисунок Т.35 – Окно редактирование данных с выведенной таблицей «Медицинские работники»

Выбранная ячейка для редактирования данных таблицы «Медицинские работники» представлена рисунке Т.36.

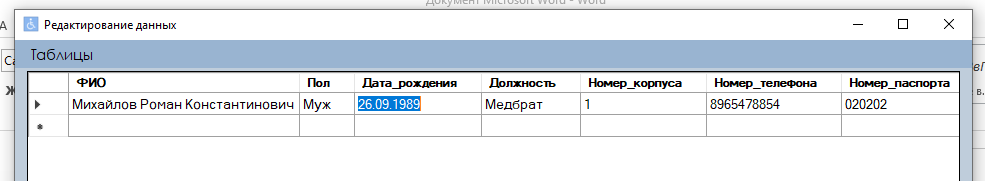


Рисунок Т.36 – Выбранная ячейка для редактирования данных таблицы «Медицинские работники»

Выбранная ячейка таблицы «Медицинские работники» с внесенными изменениями представлена на рисунке Т.37.

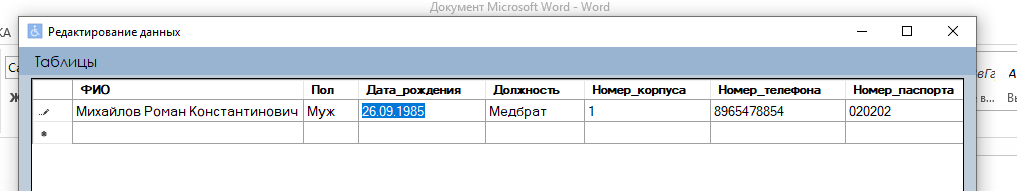


Рисунок Т.37 – Выбранная ячейка таблицы «Медицинские работники» с внесенными изменениями

Кнопка для сохранения внесенных изменений представлена на рисунке Т.38.

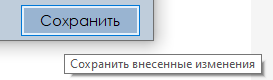


Рисунок Т.38 – Кнопка для сохранения внесенных изменений

**1.3.15 Функция возврата на главную форму**

Осуществляет возврат на основную форму программы по нажатию кнопки «Назад».

Кнопка для возврата на главную форму представлена на рисунке Т.39.

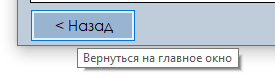


Рисунок Т.39 – Кнопка для возврата на главную форму

**1.3.16 Функция поиска записей**

Осуществляет поиск нужной записи с помощью специального поля для поиска, в которое пользователь вводит нужную ему информацию (ФИО проживающего, ФИО родственника, ФИО медицинского работника или номер корпуса) и в таблице подсвечиваются нужные данные.

Пример работы поиска представлен на рисунке Т.40.

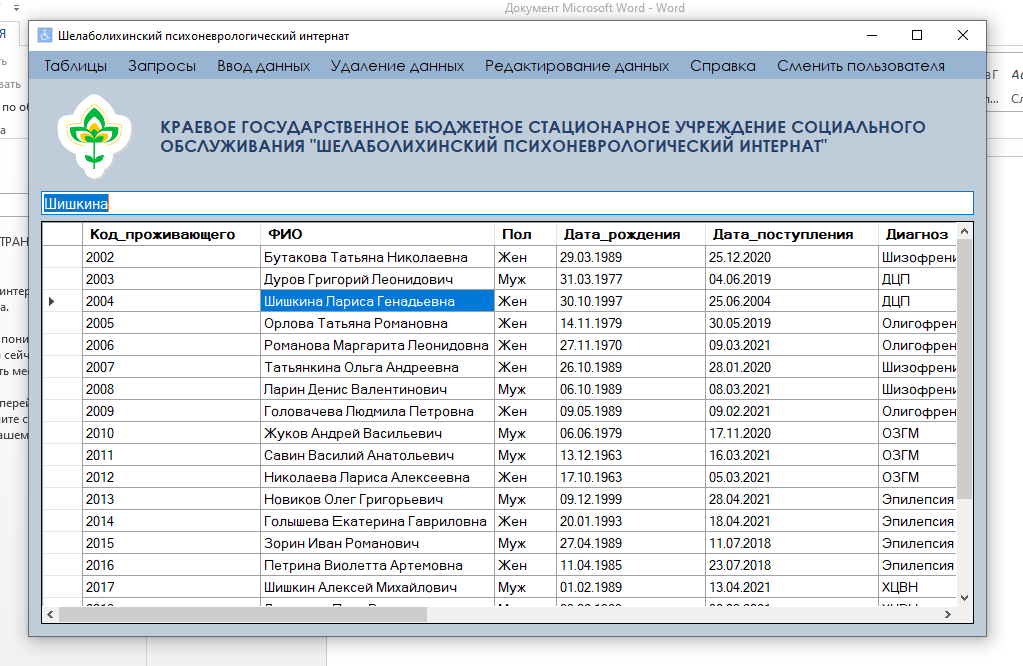


Рисунок Т.40 – Пример работы поиска

**1.3.17 Функция отображения справки**

Осуществляет вывод окна справки, содержащее руководство оператора и руководство программиста, на экран по нажатию на вкладку «Справка».

Окно справки с открытой вкладкой руководства оператора представлено на рисунке Т.41.

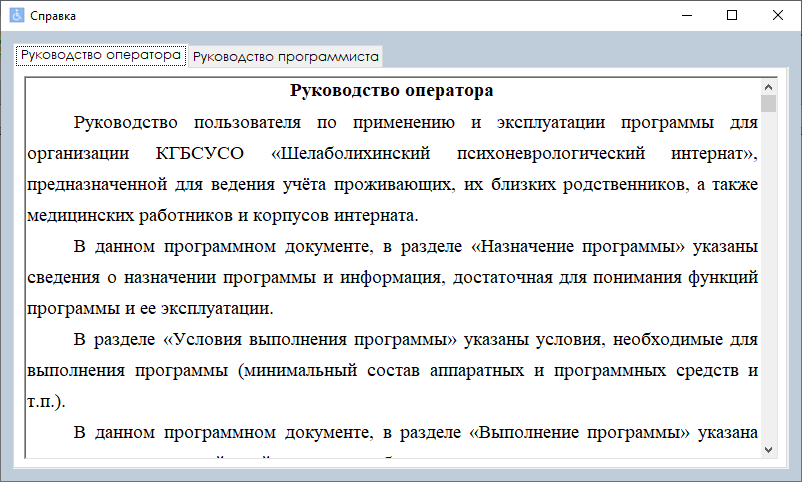


Рисунок Т. 41 – Окно справки с открытой вкладкой руководства оператора

Окно справки с открытой вкладкой руководства программиста представлено на рисунке Т.42.

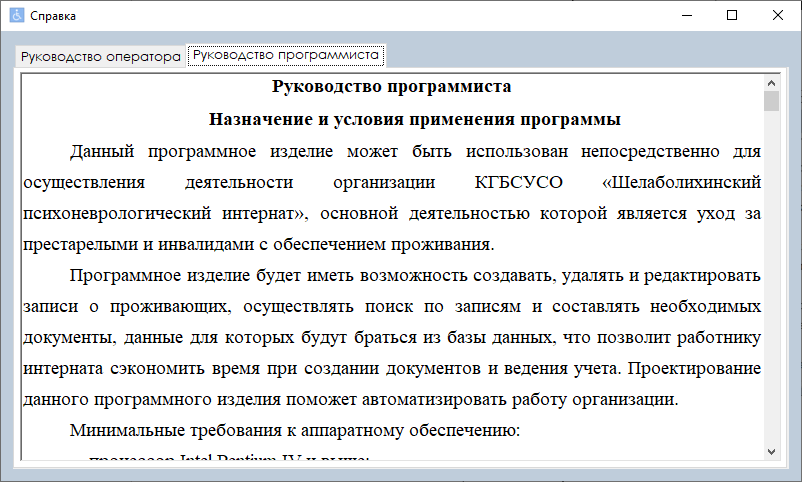


Рисунок Т.42 – Окно справки с открытой вкладкой руководства программиста

**1.3.18 Функция смены пользователя**

Осуществляет выход из аккаунта на окно авторизации по нажатию на кнопку «Сменить пользователя» на главной форме программы.

Кнопка «Сменить пользователя» представлена на рисунке Т.43.

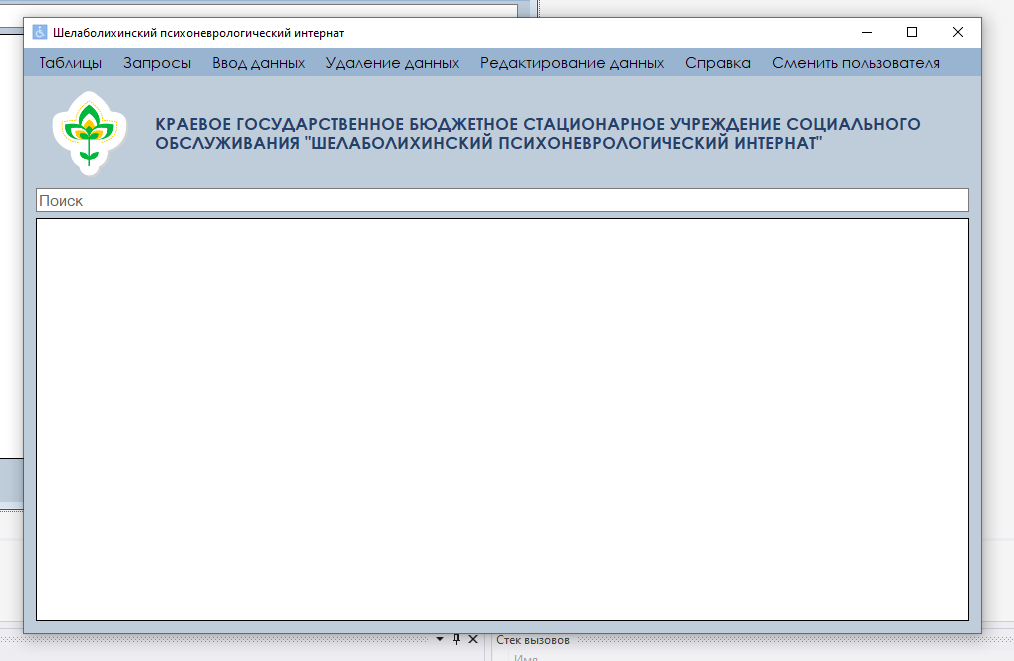


Рисунок Т.43 – Кнопка «Сменить пользователя»

2. Условия выполнения программы

**2.1 Минимальный состав аппаратных средств**

Минимальный состав используемых технических (аппаратных) средств:

* процессор Intel Pentium IV и выше;
* оперативная память 512 Мб и выше;
* видеокарта AGP/PCI Express 64 Мб и выше;
* свободное пространство на диске 20 Мб.

**2.2 Минимальный состав программных средств**

Системные программные средства, используемые программой «Шелаболихинский психоневрологический интернат», должны быть представлены локализованной версией операционных систем Windows 7/8/8.1/10.

Также для функционирования программы на ПК необходимо предустановленное программное обеспечение стороннего разработчика, версия .NETFramework v4.7.2 и выше и Microsoft SQL Server 2012 и выше.

**2.3 Требования к персоналу (пользователю)**

Конечный пользователь программы (оператор) должен обладать практическими навыками работы с пользовательским интерфейсом операционной системы.

**3. Выполнение программы**

**3.1 Загрузка и запуск программы**

Запуск программы производится по двойному нажатию левой клавиши мыши, нажатию правой клавишей мыши и выборе в контекстном меню пункта «Открыть» на ярлыке программного продукта «Шелаболихинский психоневрологический интернат».

**3.2 Выполнение программы**

**3.2.1 Выполнение функции добавления записи проживающего**

По нажатию на кнопку «Добавить» произойдёт добавление записи в таблицу «Проживающие» базы данных (рисунок Т.44).

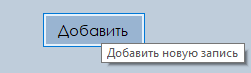


Рисунок Т.44– Кнопка «Добавить»

**3.2.2 Выполнение функции отмены добавления записи проживающего**

По нажатию на кнопку «Отмена» происходит очистка полей окна добавление записи в таблицу «Проживающие» базы данных (рисунок Т.45).

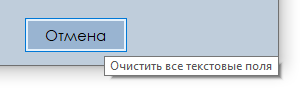


Рисунок Т.45 – Кнопка «Отмена»

**3.2.3 Выполнение функции удаления записи проживающего**

По нажатию на кнопку «Удалить запись» (рисунок Т.46) производит удаление выбранной записи из таблицы «Проживающие» базы данных.

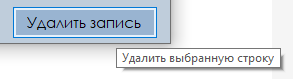


Рисунок Т.46 – Кнопка «Удалить запись»

**3.2.4 Выполнение функции** **редактирования записи проживающего**

По нажатию на кнопку «Сохранить» (рисунок Т.47) происходит изменение данных выбранной записи в таблице «Проживающие» базы данных.

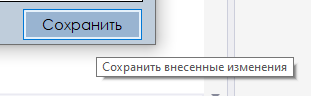


Рисунок Т.47 – Кнопка «Сохранить»

**3.2.5 Выполнение функции добавления записи родственника проживающего**

По нажатию на кнопку «Добавить» произойдёт добавление записи в таблицу «Данные о родственниках» базы данных (рисунок Т.48).

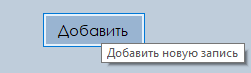


Рисунок Т.48– Кнопка «Добавить»

**3.2.6 Выполнение функции отмены добавления записи родственника проживающего**

По нажатию на кнопку «Отмена» происходит очистка полей окна добавление записи в таблицу «Данные о родственниках» базы данных (рисунок Т.49).

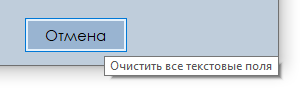


Рисунок Т.49 – Кнопка «Отмена»

**3.2.7 Выполнение функции удаления записи родственника проживающего**

По нажатию на кнопку «Удалить запись» (рисунок Т.50) производит удаление выбранной записи из таблицы «Данные о родственниках» базы данных.

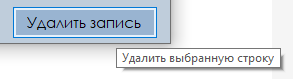


Рисунок Т.50 – Кнопка «Удалить запись»

**3.2.8 Выполнение функции редактирования записи родственника проживающего**

По нажатию на кнопку «Сохранить» (рисунок Т.51) происходит изменение данных выбранной записи в таблице «Данные о родственниках» базы данных.

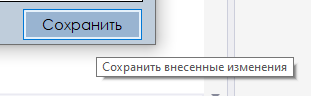


Рисунок Т.51 – Кнопка «Сохранить»

**3.2.9 Выполнение функции редактирования записи корпуса**

По нажатию на кнопку «Сохранить» (рисунок Т.52) происходит изменение данных выбранной записи в таблице «Корпуса» базы данных.

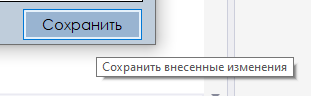


Рисунок Т.52 – Кнопка «Сохранить»

**3.2.10 Выполнение функции добавления записи медицинского работника**

По нажатию на кнопку «Добавить» произойдёт добавление записи в таблицу «Медицинские работники» базы данных (рисунок Т.53).

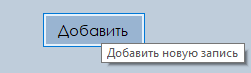


Рисунок Т.53 – Кнопка «Добавить»

**3.2.11 Выполнение функции отмены добавления записи медицинского работника**

По нажатию на кнопку «Отмена» происходит очистка полей окна добавление записи в таблицу «Медицинские работники» базы данных (рисунок Т.54).

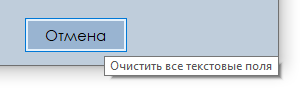


Рисунок Т.54 – Кнопка «Отмена»

**3.2.12 Выполнение функции удаления записи медицинского работника**

По нажатию на кнопку «Удалить запись» (рисунок Т.55) производит удаление выбранной записи из таблицы «Медицинские работники» базы данных.

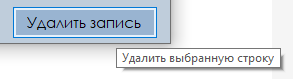


Рисунок Т.55 – Кнопка «Удалить запись»

**3.2.13 Выполнение функции редактирования записи медицинского работника**

По нажатию на кнопку «Сохранить» (рисунок Т.56) происходит изменение данных выбранной записи в таблице «Медицинские работники» базы данных.

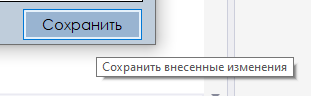


Рисунок Т.56 – Кнопка «Сохранить»

**3.2.14 Выполнение функции отображения руководства оператора**

По нажатию на кнопку «Справка» (Рисунок Т.57) и далее в окне справки выбора вкладки «Руководство оператора» осуществляется вывод текста руководства оператора на экран (Рисунок Т.58).

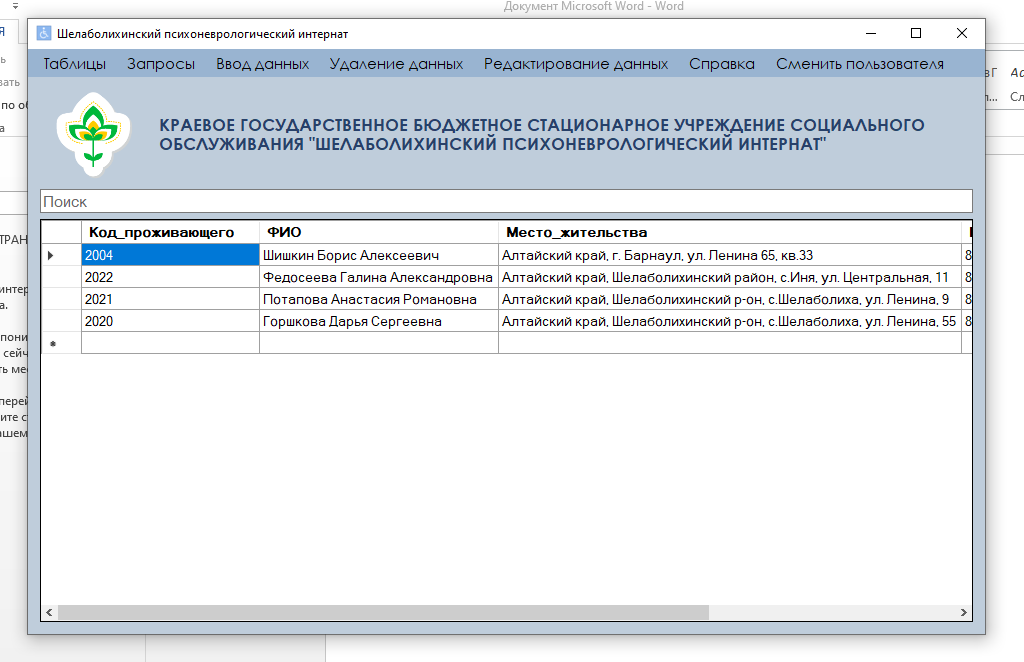


Рисунок Т.57 – Кнопка «Справка»

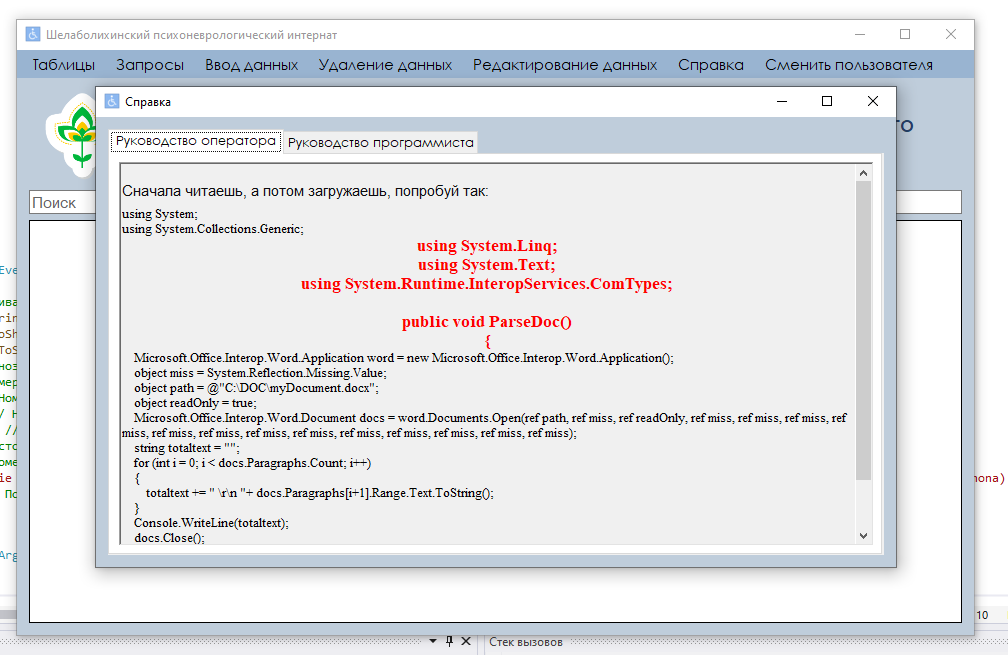


Рисунок Т.58 – Вкладка «Руководство оператора»

**3.2.15 Выполнение функции отображения руководства программиста**

По нажатию на кнопку «Справка» (Рисунок Т.57) и далее в окне справки выбора вкладки «Руководство программиста» осуществляется вывод текста руководства оператора на экран (Рисунок Т.59).

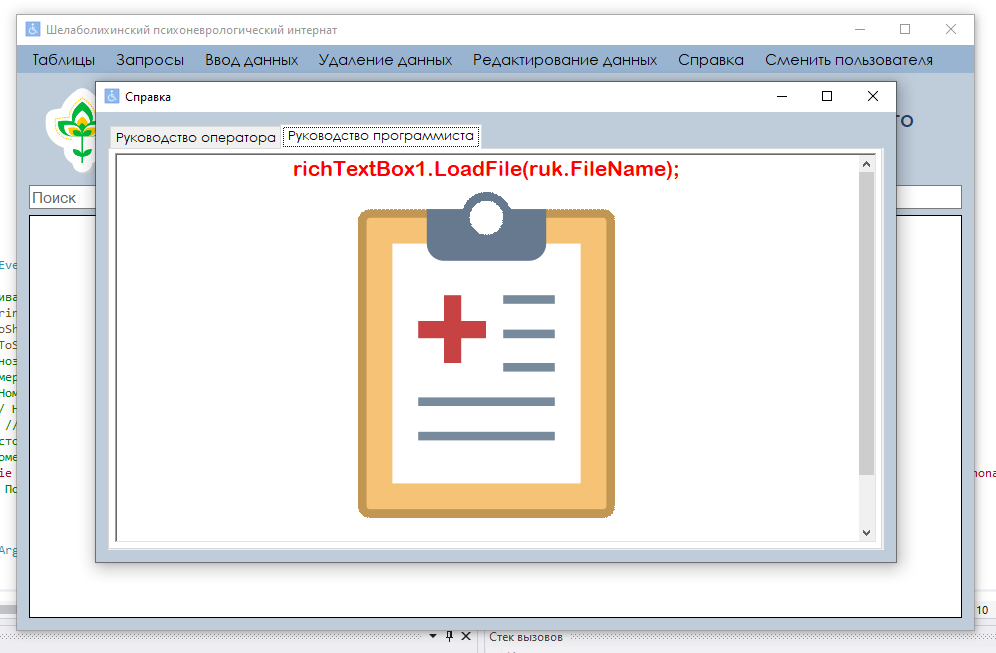


Рисунок Т.59 – Вкладка «Руководство программиста»

**3.3 Завершение работы программы**

Закрытие программы осуществляется при нажатии системной кнопки «Закрыть».

**4. Сообщения оператору**

**4.1 Справка**

Программа выдает справочный материал о программе, при открытии вкладки «Справка» показанный на рисунках Т.60 и Т.61.

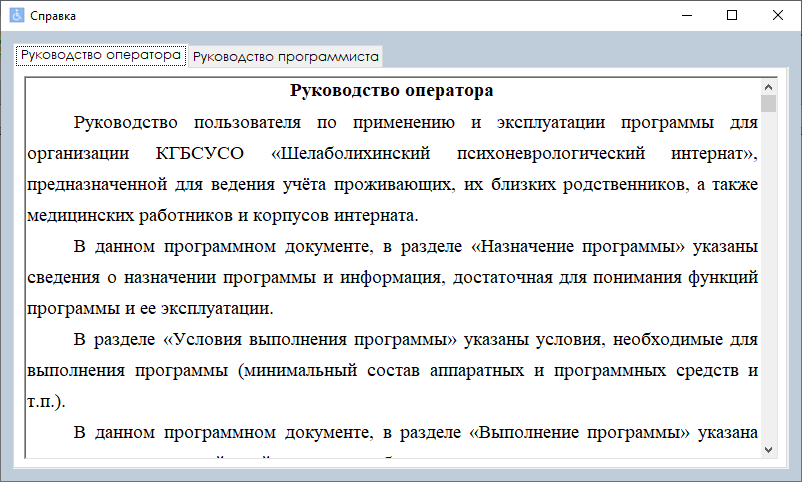


Рисунок Т.60 – Справка

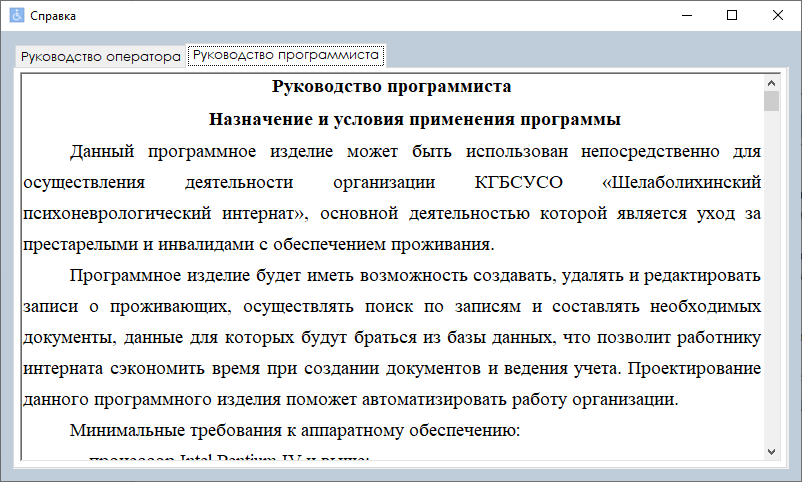


Рисунок Т.61 – Справка

**Приложение** **У**

**Руководство программиста**

**Назначение и условия применения программы**

Данный программное изделие может быть использован непосредственно для осуществления деятельности организации КГБСУСО «Шелаболихинский психоневрологический интернат», основной деятельностью которой является уход за престарелыми и инвалидами с обеспечением проживания.

Программное изделие будет иметь возможность создавать, удалять и редактировать записи о проживающих, осуществлять поиск по записям и составлять необходимых документы, данные для которых будут браться из базы данных, что позволит работнику интерната сэкономить время при создании документов и ведения учета. Проектирование данного программного изделия поможет автоматизировать работу организации.

Минимальные требования к аппаратному обеспечению:

* процессор Intel Pentium IV и выше;
* оперативная память 512 Мб и выше;
* видеокарта (любая) 64 Мб и выше;
* свободное пространство на диске от 12 Мб;
* видеомонитор с разрешением 1280x720 и более;
* клавиатура;
* мышь;

Минимальные системные требования:

* операционная система Windows 7/8/8.1/10;
* .NETFramework v.4.7.2 и выше;
* Microsoft SQL Server 2012 и выше.

**Характеристика программы**

В тестовом режиме был произведен запрос к форме добавления проживающего и введены данные в текстовые поля, что позволило оценить наглядно загруженность центрального процессора (ЦП) и использование выделенной (виртуальной) памяти с помощью «Диспетчера задач», как изображено на рисунках У.1 и У.2.

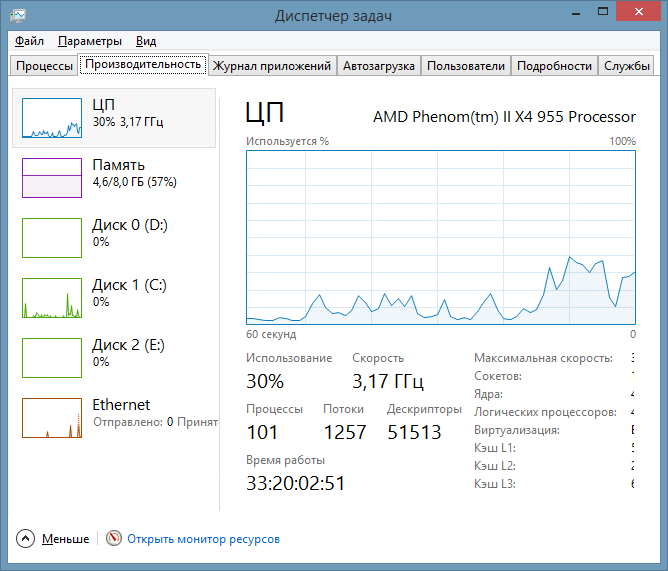


Рисунок У.1 – Загрузка центрального процессора

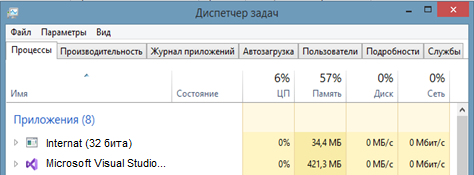


Рисунок У.2 – Выделенная память и время загрузки

**Обращение к программе**

Запуск приложения возможен по щелчку на иконке самой программы, находящейся в специальном каталоге. После запуска приложения на экране отображается главное окно, с помощью которого можно перемещаться между вкладками приложения и выполнять функции, описанными в разделе 4 руководства пользователя.

Запустить данную программу можно непосредственно через исполнительный файл с разрешением \*.exe. Для этого требуется открыть файл проекта Program.exe, находящемся в каталоге программы. Далее запустится само программное изделие.

Выход из приложения возможен при нажатии на системную кнопку «Закрыть».

**Полный перечень модулей и компонентов**

Весь программный продукт состоит из 8 форм: Form 1, Form 2, Form 3, Form 4, Form 5, Form 7, Form 8, Form 9.

Form1 является окном авторизации в программе, которое содержит два текстовых поля для ввода логина и пароля и кнопку «Войти».

Form2 является основным окном программы, которое содержит вкладки «Таблицы», «Запросы», «Ввод данных» и кнопки «Удаление данных», «Редактирование данных», «Справка» и «Сменить пользователя». Form2 является окном для вывода всех таблиц базы данных, вывода проживающих с фильтрацией по диагнозу, а также для осуществления поиска по записям таблиц.

Form3 является окном добавления нового проживающего, содержит текстовые поля соответствующие таблице «Проживающие», кнопку добавления новой записи и кнопку отмены.

Form4 является окном добавления нового родственника проживающего, содержит текстовые поля соответствующие таблице «Данные о родственниках», кнопку добавления новой записи и кнопку отмены.

Form5 является окном добавления медицинского работника, содержит текстовые поля соответствующие таблице «Медицинские работники», кнопку добавления новой записи и кнопку отмены.

Form7 является окном удаления данных, которое содержит в себе меню для вывода таблиц базы данных, поле для поиска по записям, кнопку «Удалить запись» и кнопку «Назад» для возврата на главное окно программы.

Form8 является окном редактирования данных, которое содержит в себе меню для вывода таблиц базы данных, кнопку «Сохранить» и кнопку «Назад» для возврата на главное окно программы.

Form9 является окном справки, содержащее руководство оператора и руководство программиста.

Вся основная работа программы управляется кодом, содержащимся в модулях.

В состав данного программного продукта входят следующие модули:

1. Модуль вывода данных о проживающих. Модуль осуществляет вывод данных на главное окно программы из таблицы базы данных «Проживающие».
2. Модуль вывода данных о родственниках проживающих. Модуль осуществляет вывод данных на главное окно программы из таблицы базы данных «Данные о родственниках».
3. Модуль вывода данных о корпусах интерната. Модуль осуществляет вывод данных на главное окно программы из таблицы базы данных «Корпуса».
4. Модуль вывода данных о медицинских работниках. Модуль осуществляет вывод данных на главное окно программы из таблицы базы данных «Медицинские работники».
5. Модуль добавления нового проживающего. Модуль осуществляет добавление записи проживающего в соответствующую таблицу базы данных после заполнения всех текстовых полей и нажатия кнопки «Добавить».
6. Модуль добавления нового родственника проживающего. Модуль осуществляет добавление записи родственника проживающего в соответствующую таблицу базы данных после заполнения всех текстовых полей и нажатия кнопки «Добавить».
7. Модуль добавления нового медицинского работника. Модуль осуществляет добавление записи медицинского работника в соответствующую таблицу базы данных после заполнения всех текстовых полей и нажатия кнопки «Добавить».
8. Для очистки текстовых полей существует кнопка «Отмена», которая при нажатии осуществляет очистку всех текстовых полей окна.
9. Модуль удаления данных. Модуль осуществляет удаление выбранной записи из таблицы базы данных после нажатия кнопки «Удалить запись».
10. Модуль редактирования данных. Модуль осуществляет изменение выбранной записи из таблицы базы данных после внесения и нажатия кнопки «Сохранить».
11. Модуль поиска данных. Модуль осуществляет поиска по строкам таблицы базы данных с помощью специального текстового поля. При обнаружении совпадения строка таблицы подсвечивается.
12. Модуль справки. Осуществляет вывод справки при нажатии на кнопку «Справка» на главном окне программы, содержит в себе руководство оператора и руководство программиста.

**Сообщение пользователю**

При неверно введенном ввода логина и/или пароля выводится сообщение об ошибке, представленное на рисунке У.3.

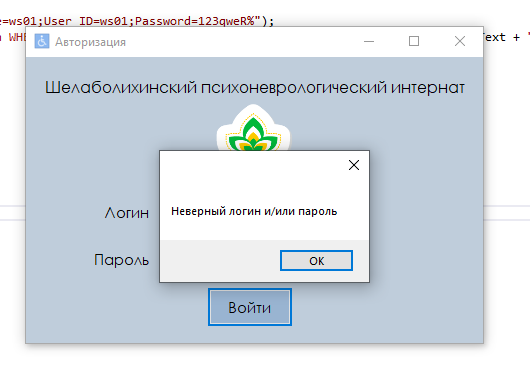


Рисунок У.3 – Сообщение об ошибке

Если произошел сбой при установке подключении программы к базе данных, то выводиться сообщение об ошибке, показанное на рисунке У.4

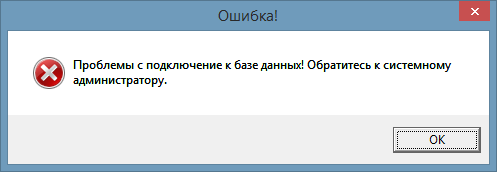


Рисунок У.4 – Сообщение об ошибке.

Если при добавлении новой записи пользователь ввел все данные правильно и нажал кнопку «Добавить», то по выполнению действия выводится сообщение о том, что данные, введенные ранее, успешно сохранены.

Сообщение показано на рисунке У.5.

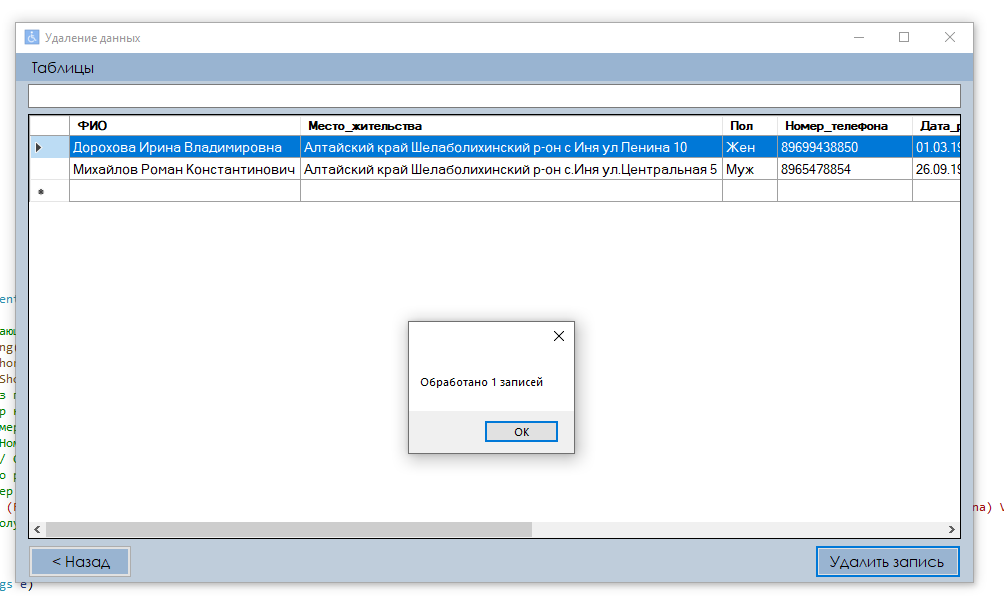


Рисунок У.5 – Сообщение об успешном выполнении

Если после внесения изменений в запись в окне «Редактирование данных» пользователь нажал кнопку «Сохранить», то по выполнению действия выводится сообщение о том, что данные, введенные ранее, успешно сохранены.

Сообщение показано на рисунке У.6

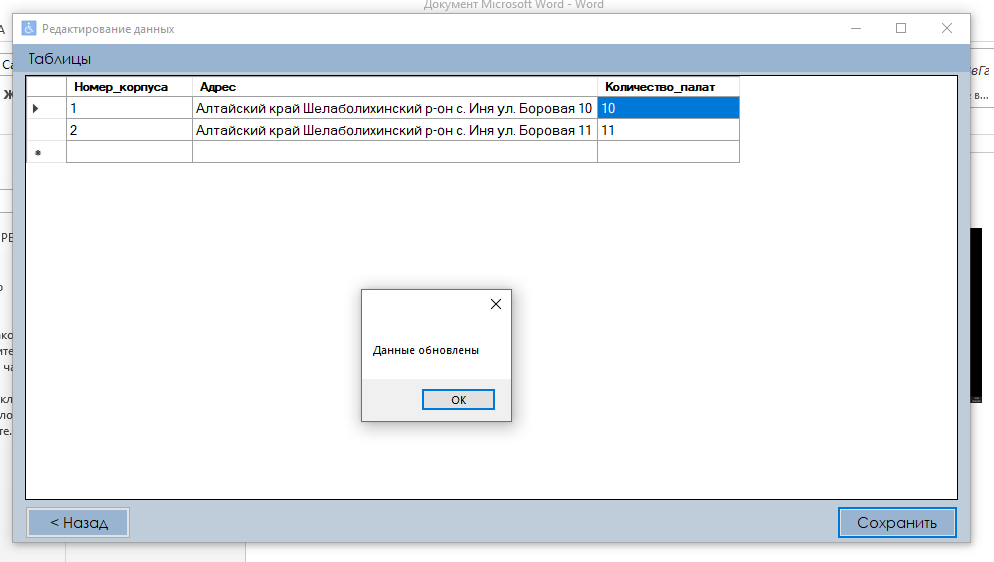


Рисунок У.6 – Сообщение об успешном выполнении

**Приложение Ф**

# Программный код

**Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WF\_SQLclass

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

// Авторизация в программе

SqlConnection con = new SqlConnection(@"Data Source=WSS\SQLEXPRESS;Database=ws01;User ID=ws01;Password=123qweR%");

SqlDataAdapter sda = new SqlDataAdapter("SELECT COUNT (\*) FROM Avtorization WHERE login = '" + textBox1.Text + "' AND password = '" + textBox2.Text + "'", con);

DataTable dt = new DataTable();

sda.Fill(dt);

if (dt.Rows[0][0].ToString() == "1")

{

Form2 F2 = new Form2(); // Переход на главную форму приложения

F2.Show();

this.Hide();

}

else

{

// Вывод сообщения об ошибочно введенном логине и/или пароле

MessageBox.Show("Неверный логин и/или пароль");

}

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

e.Cancel = MessageBox.Show("Вы действительно хотите выйти?", "Выход", MessageBoxButtons.YesNo) == DialogResult.No; // Вывод вопросительного сообщения

}

}

}

}

**Form 2.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WF\_SQLclass

{

public partial class Form2 : Form

{

int SearchI = 0;

public Form2()

{

InitializeComponent();

}

// ОТОБРАЖЕНИЕ ОСНОВНЫХ ТАБЛИЦ

private void группыToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы ПРОЖИВАЮЩИЕ

string select = "SELECT ID\_Pr AS Код\_проживающего, FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie ORDER BY FIO"; // Запрос на получение данных, вывод таблицы ПРОЖИВАЮЩИЕ

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные. dataGridView1 должен присутствовать на форме

dataGridView1.Columns[0].Visible = false;

SearchI = 1;

}

private void студентыToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы ДАННЫЕ О РОДСТВЕННИКАХ

string select = "SELECT ID\_rodstvennika, FIO AS ФИО, Mesto\_gitelstva AS Место\_жительства, Nomer\_telephona AS Номер\_телефона, Email AS Электронная\_почта, ID\_Pr AS Код\_проживающего FROM Dannie\_o\_rodstvennikah ORDER BY FIO";

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные. dataGridView1 должен присутствовать на форме

dataGridView1.Columns[0].Visible = false;

SearchI = 2;

}

private void сотрудникиТрудоваяДеятельностьToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы КОРПУСА

string select = "SELECT ID\_korpusa AS Номер\_корпуса, Adres AS Адрес, Kolvo\_palat AS Количество\_палат FROM Korpusa ORDER BY ID\_korpusa";

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные. dataGridView1 должен присутствовать на форме

SearchI = 3;

}

private void сотрудникиЛичныеДанныеToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы МЕДИЦИНСКИЕ РАБОТНИКИ

string select = "SELECT ID\_med\_rabotnika, FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Dolgnost AS Должность, ID\_korpusa AS Номер\_корпуса, Nomer\_telephona AS Номер\_телефона, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Mesto\_gitelstva AS Место\_жительства FROM Med\_rabotniki ORDER BY FIO";

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные. dataGridView1 должен присутствовать на форме

dataGridView1.Columns[0].Visible = false;

SearchI = 4;

}

private void button1\_Click(object sender, EventArgs e)

{

Application.Exit(); // Закрытие формы

}

private void вакансииToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Form4 F4 = new Form4(); // Форма ввода новых данных о родственнике

F4.Show();

}

private void сотрудникиТрудоваяДеятельностьToolStripMenuItem1\_Click(object sender, EventArgs e)

{

Form5 F5 = new Form5(); // Форма ввода новых данных о медицинском работнике

F5.Show();

}

private void toolStripMenuItem1\_Click(object sender, EventArgs e)

{

Form3 F3 = new Form3(); // Форма ввода проживающего

F3.Show();

}

private void удалениеДанныхToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form7 F7 = new Form7(); // Переход на форму для удаления записи

F7.Show();

this.Hide();

}

private void выводПроживающихСДиагнозомШизофренияToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом шизофрения)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'Шизофрения' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void выводПроживающихСДиагнозомОлигофренияToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом Олигофрения)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'Олигофрения' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void выводПроживающихСДиагнозомОЗГМToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом ОЗГМ)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'ОЗГМ' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void выводПроживающихСДиагнозомЭпилепсияToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом Эпилепсия)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'Эпилепсия' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void выводПроживающихСДиагнозомХЦВНToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом ХЦВНТ)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'ХЦВН' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void выводПроживающихСДиагнозомДЦПToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом ДЦП)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'ДЦП' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void выводПроживающихСДиагнозомСосудистаToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом сосудистая деменция)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'Сосудистая деменция' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void выводПроживающихСДиагнозомДеменцияToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Осуществление вывода запроса (Проживающие с диагнозом деменция)

string select = "SELECT FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie WHERE Diagnoz = 'Деменция' ORDER BY FIO";

DataTable dt = new DataTable();

SQLquery.Select(select, ref dt);

dataGridView1.DataSource = dt;

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

// Поиск по записям

int k = 0;

if (SearchI == 1) k = 1;

if (SearchI == 2) k = 1;

if (SearchI == 3) k = 0;

if (SearchI == 4) k = 1;

for (int i = 0; i < dataGridView1.RowCount; i++)

if (dataGridView1[k, i].FormattedValue.ToString().

Contains(textBox1.Text.Trim()))

{

dataGridView1.CurrentCell = dataGridView1[1, i];

return;

}

}

private void Form2\_FormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

e.Cancel = MessageBox.Show("Вы действительно хотите выйти?", "Выход", MessageBoxButtons.YesNo) == DialogResult.No; // Вывод вопросительного сообщения

}

}

private void справкаToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form8 F8 = new Form8(); // Переход на форму справки

F8.Show();

}

private void редактированиеДанныхToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form9 F9 = new Form9(); // Переход на форму редактирования данных

F9.Show();

this.Hide();

}

private void сменитьПользователяToolStripMenuItem\_Click(object sender, EventArgs e)

{

Form1 F1 = new Form1(); // Переход на форму авторизации

F1.Show();

this.Hide();

}

private void textBox1\_MouseClick(object sender, MouseEventArgs e)

{

textBox1.Text = "";

}

}

}

**Form3.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WF\_SQLclass

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

string FIO = textBox1.Text; // ФИО проживающего

string Pol = comboBox2.SelectedItem.ToString(); // Пол проживающего

string DataRog = dateTimePicker1.Value.ToShortDateString(); // Дата рождения

string DataPost = dateTimePicker2.Value.ToShortDateString(); // Дата поступления проживающего

string Diagnoz = textBox8.Text; // Диагноз проживающего

string IDkorpusa = textBox7.Text; // Номер корпуса

string Nomerpalati = textBox6.Text; // Номер палаты

string Nomerpasporta = textBox5.Text; // Номер паспорта

string Seriyapasporta = textBox12.Text; // Серия паспорта

string Mestorog = textBox11.Text; // Место рождения проживающего

string Telephone = textBox10.Text; // Номер телефона проживающего

string insert = "INSERT INTO Progivayushie (FIO, Pol, Data\_rog, Data\_postupleniya, Diagnoz, ID\_korpusa, Nomer\_palati, Nomer\_pasporta, Seriya\_pasporta, Mesto\_rogdeniya, Nomer\_telephona) VALUES ('" + FIO + "','" + Pol + "', '" + DataRog + "', '" + DataPost + "','" + Diagnoz + "', '" + IDkorpusa + "','" + Nomerpalati + "','" + Nomerpasporta + "', '" + Seriyapasporta + "', '" + Mestorog + "', '" + Telephone + "')"; // Запрос добавления проживающего

SQLquery.Insert(insert); // Получаем данные из базы данных

}

private void button2\_Click(object sender, EventArgs e)

{

// Функция очистки всех текстовых полей

textBox1.Clear();

comboBox2.SelectedIndex = -1;

dateTimePicker1.Value = DateTime.Now;

dateTimePicker2.Value = DateTime.Now;

textBox8.Clear();

textBox7.Clear();

textBox6.Clear();

textBox5.Clear();

textBox12.Clear();

textBox11.Clear();

textBox10.Clear();

}

private void button1\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button1, "Добавить новую запись"); // Всплывающее сообщение

}

private void button2\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button2, "Очистить все текстовые поля"); // Всплывающее сообщение

}

private void Form3\_FormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

e.Cancel = MessageBox.Show("Вы действительно хотите прервать добавление?", "Выход", MessageBoxButtons.YesNo) == DialogResult.No; // Вывод вопросительного сообщения

}

}

}

}

**Form4.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WF\_SQLclass

{

public partial class Form4 : Form

{

public Form4()

{

InitializeComponent();

// заполнение combobox фамилией проживающего

string select = "SELECT ID\_Pr, FIO FROM Progivayushie"; // Запрос на получение данных, вывод таблицы ПРОЖИВАЮЩИЕ

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

comboBox1.DataSource = dt;

comboBox1.DisplayMember = dt.Columns[1].ColumnName;

comboBox1.ValueMember = dt.Columns[0].ColumnName;

}

private void button1\_Click\_1(object sender, EventArgs e)

{

string FIO = textBox1.Text; // ФИО родственника

string MestoGit = textBox2.Text; // Место жительства родственника

string Telephone = textBox3.Text; // Номер телефона родственника

string Email = textBox4.Text; // Электронная почта родственника

long idep = Convert.ToInt64(comboBox1.SelectedValue);// Код проживающего

string insert = "INSERT INTO Dannie\_o\_rodstvennikah (FIO, Mesto\_gitelstva, Nomer\_telephona, Email, ID\_Pr) VALUES ('" + FIO + "','" + MestoGit + "', '" + Telephone + "', '" + Email + "'," + idep + ")"; // Запрос добавления данных о родственниках

SQLquery.Insert(insert); // Получаем данные из базы данных

}

private void button2\_Click(object sender, EventArgs e)

{

// Функция очистки всех текстовых полей

textBox1.Clear();

textBox2.Clear();

textBox3.Clear();

textBox4.Clear();

comboBox1.SelectedIndex = -1;

}

private void button1\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button1, "Добавить новую запись"); // Всплывающее сообщение

}

private void button2\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button2, "Очистить все текстовые поля"); // Всплывающее сообщение

}

private void Form4\_FormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

e.Cancel = MessageBox.Show("Вы действительно хотите прервать добавление?", "Выход", MessageBoxButtons.YesNo) == DialogResult.No; // Вывод вопросительного сообщения

}

}

}

}

**Form5.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WF\_SQLclass

{

public partial class Form5 : Form

{

public Form5()

{

InitializeComponent();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

string FIO = textBox1.Text; // ФИО медработника

string Pol = comboBox1.SelectedItem.ToString(); // Пол медработника

string DataRog = dateTimePicker1.Value.ToShortDateString(); // Дата рождения медработника

string Dolgnost = textBox4.Text; // Должность медработника

string IDkorpusa = textBox5.Text; // Код корпуса

string Telephone = textBox7.Text; // Должность медработника

string Nomerpasporta = textBox8.Text; // Код корпуса

string Seriyapasporta = textBox9.Text; // Код корпуса

string Mestorogdeniya = textBox10.Text; // Код корпуса

string Mestogit = textBox11.Text; // Код корпуса

string insert = "INSERT INTO Med\_rabotniki (FIO, Pol, Data\_rog, Dolgnost, ID\_korpusa, Nomer\_telephona, Nomer\_pasporta, Seriya\_pasporta, Mesto\_rogdeniya, Mesto\_gitelstva) VALUES ('" + FIO + "','" + Pol + "', '" + DataRog + "', '" + Dolgnost + "','" + IDkorpusa + "','" + Telephone + "','" + Nomerpasporta + "','" + Seriyapasporta + "','" + Mestorogdeniya + "','" + Mestogit + "')"; // Запрос добавления медработника

SQLquery.Insert(insert); // Получаем данные из базы данных

}

private void button2\_Click(object sender, EventArgs e)

{

// Функция очистки всех текстовых полей

textBox1.Clear();

comboBox1.SelectedIndex = -1;

dateTimePicker1.Value = DateTime.Now;

textBox4.Clear();

textBox5.Clear();

textBox7.Clear();

textBox8.Clear();

textBox9.Clear();

textBox10.Clear();

textBox11.Clear();

}

private void button1\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button1, "Добавить новую запись"); // Всплывающее сообщение

}

private void button2\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button2, "Очистить все текстовые поля"); // Всплывающее сообщение

}

private void Form5\_FormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

e.Cancel = MessageBox.Show("Вы действительно хотите прервать добавление?", "Выход", MessageBoxButtons.YesNo) == DialogResult.No; // Вывод вопросительного сообщения

}

}

}

}

**Form7.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WF\_SQLclass

{

public partial class Form7 : Form

{

int DelN = 0;

int SearchI = 0;

public Form7()

{

InitializeComponent();

}

// Запрос на получение данных, вывод таблицы ПРОЖИВАЮЩИЕ

private void группыToolStripMenuItem\_Click(object sender, EventArgs e)

{

string select = "SELECT ID\_Pr, FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie"; // Запрос на получение данных, вывод таблицы ПРОЖИВАЮЩИЕ

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные. dataGridView1 должен присутствовать на форме

DelN = 1;

dataGridView1.Columns[0].Visible = false;

SearchI = 1;

}

// Запрос на получение данных, вывод таблицы ДАННЫЕ О РОДСТВЕННИКАХ

private void студентыToolStripMenuItem\_Click(object sender, EventArgs e)

{

string select = "SELECT ID\_rodstvennika, FIO AS ФИО, Mesto\_gitelstva AS Место\_жительства, Nomer\_telephona AS Номер\_телефона, Email AS Электронная\_почта, ID\_Pr AS Номер\_проживающего FROM Dannie\_o\_rodstvennikah";

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные. dataGridView1 должен присутствовать на форме

DelN = 2;

dataGridView1.Columns[0].Visible = false;

SearchI = 2;

}

// Запрос на получение данных, вывод таблицы МЕДИЦИНСКИЕ РАБОТНИКИ

private void сотрудникиЛичныеДанныеToolStripMenuItem\_Click(object sender, EventArgs e)

{

string select = "SELECT ID\_med\_rabotnika, FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Dolgnost AS Должность, ID\_korpusa AS Номер\_корпуса, Nomer\_telephona AS Номер\_телефона, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Mesto\_gitelstva AS Место\_жительства FROM Med\_rabotniki ";

DataTable dt = new DataTable(); // Создаем объект DataTable

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные. dataGridView1 должен присутствовать на форме

DelN = 4;

dataGridView1.Columns[0].Visible = false;

SearchI = 4;

}

private void button1\_Click(object sender, EventArgs e)

{

Form2 F2 = new Form2(); // Переход на основную форму приложения

F2.Show();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

// Удаление записей из таблиц

string p = dataGridView1[0, dataGridView1.CurrentRow.Index].Value.ToString();

string delete = "";

if (DelN == 1) delete = "DELETE FROM Progivayushie WHERE ID\_Pr = " + p;

if (DelN == 2) delete = "DELETE FROM Dannie\_o\_rodstvennikah WHERE ID\_rodstvennika = " + p;

if (DelN == 3) delete = "DELETE FROM Korpusa WHERE ID\_korpusa = " + p;

if (DelN == 4) delete = "DELETE FROM Med\_rabotniki WHERE ID\_med\_rabotnika = " + p;

SQLquery.Insert(delete);

}

private void textBox1\_TextChanged\_1(object sender, EventArgs e)

{

// Поиск по таблицам

int k = 0;

if (SearchI == 1) k = 1;

if (SearchI == 2) k = 1;

if (SearchI == 3) k = 0;

if (SearchI == 4) k = 1;

for (int i = 0; i < dataGridView1.RowCount; i++)

if (dataGridView1[k, i].FormattedValue.ToString().

Contains(textBox1.Text.Trim()))

{

dataGridView1.CurrentCell = dataGridView1[1, i];

return;

}

}

private void Form7\_FormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

e.Cancel = MessageBox.Show("Вы действительно хотите выйти?", "Выход", MessageBoxButtons.YesNo) == DialogResult.No; // Вывод вопросительного сообщения

}

}

private void button1\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button1, "Вернуться на главное окно"); // Всплывающее сообщение

}

private void button2\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button2, "Удалить выбранную строку"); // Всплывающее сообщение

}

private void textBox1\_MouseClick(object sender, MouseEventArgs e)

{

textBox1.Text = "";

}

}

}

**Form8.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace WF\_SQLclass

{

public partial class Form8 : Form

{

public Form8()

{

InitializeComponent();

}

// Осуществление вывода справки

private void Form8\_MouseDown(object sender, MouseEventArgs e)

{

base.Capture = false;

Message m = Message.Create(base.Handle, 0xa1, new IntPtr(2), IntPtr.Zero);

this.WndProc(ref m);

}

private void Form8\_Load(object sender, EventArgs e)

{

OpenFileDialog ruk = new OpenFileDialog();

OpenFileDialog sprv = new OpenFileDialog();

ruk.FileName = Application.StartupPath + "/Дополнения/rukoperatora.rtf";

sprv.FileName = Application.StartupPath + "/Дополнения/rukprogram.rtf";

richTextBox1.LoadFile(ruk.FileName);

richTextBox2.LoadFile(sprv.FileName);

}

}

}

**Form9.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WF\_SQLclass

{

public partial class Form9 : Form

{

DataTable dt = new DataTable(); // Создаем объект DataTable

string select = "";

public Form9()

{

InitializeComponent();

}

private void группыToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы проживающие

dt.Clear(); // Сброс таблицы

while (dt.Columns.Count > 0) dt.Columns.RemoveAt(0);

select = "SELECT ID\_Pr, FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Data\_postupleniya AS Дата\_поступления, Diagnoz AS Диагноз, ID\_korpusa AS Номер\_корпуса, Nomer\_palati AS Номер\_палаты, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Nomer\_telephona AS Номер\_телефона FROM Progivayushie"; // Запрос на получение данных, вывод таблицы ПРОЖИВАЮЩИЕ

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные

dataGridView1.Columns[0].Visible = false;

}

private void студентыToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы данные о родственниках

dt.Clear(); // Сброс таблицы

while (dt.Columns.Count > 0) dt.Columns.RemoveAt(0);

select = "SELECT ID\_rodstvennika, FIO AS ФИО, Mesto\_gitelstva AS Место\_жительства, Nomer\_telephona AS Номер\_телефона, Email AS Электронная\_почта, ID\_Pr AS Номер\_проживающего FROM Dannie\_o\_rodstvennikah";

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные

dataGridView1.Columns[0].Visible = false;

}

private void корпусаToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы КОРПУСА

dt.Clear(); // Сброс таблицы

while (dt.Columns.Count > 0) dt.Columns.RemoveAt(0);

select = "SELECT ID\_korpusa AS Номер\_корпуса, Adres AS Адрес, Kolvo\_palat AS Количество\_палат FROM Korpusa";

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные

}

private void сотрудникиЛичныеДанныеToolStripMenuItem\_Click(object sender, EventArgs e)

{

// Запрос на получение данных, вывод таблицы медицинские работники

dt.Clear(); // Сброс таблицы

while (dt.Columns.Count > 0) dt.Columns.RemoveAt(0);

select = "SELECT ID\_med\_rabotnika, FIO AS ФИО, Pol AS Пол, Data\_rog AS Дата\_рождения, Dolgnost AS Должность, ID\_korpusa AS Номер\_корпуса, Nomer\_telephona AS Номер\_телефона, Nomer\_pasporta AS Номер\_паспорта, Seriya\_pasporta AS Серия\_паспорта, Mesto\_rogdeniya AS Место\_рождения, Mesto\_gitelstva AS Место\_жительства FROM Med\_rabotniki ";

SQLquery.Select(select, ref dt); // Получаем данные из базы данных

dataGridView1.DataSource = dt; // Отображаем данные

dataGridView1.Columns[0].Visible = false;

}

private void button2\_Click(object sender, EventArgs e)

{

SQLquery.TabUpdate(select, ref dt); // Кнопка для сохранения внесенных изменений

}

private void button1\_Click(object sender, EventArgs e)

{

Form2 F2 = new Form2(); // Переход на основную форму приложения

F2.Show();

this.Hide();

}

private void button1\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button1, "Вернуться на главное окно"); // Всплывающее сообщение

}

private void button2\_MouseHover(object sender, EventArgs e)

{

toolTip1.SetToolTip(button2, "Сохранить внесенные изменения"); // Всплывающее сообщение

}

private void Form9\_FormClosing(object sender, FormClosingEventArgs e)

{

if (e.CloseReason == CloseReason.UserClosing)

{

e.Cancel = MessageBox.Show("Вы действительно хотите выйти?", "Выход", MessageBoxButtons.YesNo) == DialogResult.No; // Вывод вопросительного сообщения

}

}

}

}

**Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data;

using System.Configuration;

using System.Data.SqlClient;

namespace WF\_SQLclass

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

static class SQLquery

{

// Строка подключения. Указать свои данные: сервер, база данных, пользователь, пароль

static string connectionString = @"Data Source=WSS\SQLEXPRESS;Database=ws01;User ID=ws01;Password=123qweR%";

// Создание подключения

static SqlConnection connection = new SqlConnection(connectionString);

static public void Select(string select, ref DataTable dt) // Получение результатов запроса select в таблицу dt

{

try

{

connection.Open(); // Открываем подключение

SqlDataAdapter adapter = new SqlDataAdapter(select, connection); // Создаем объект DataAdapter

adapter.Fill(dt); // Заполняем DataTable

}

catch (SqlException ex)

{

MessageBox.Show("Ошибка подключения");

}

finally

{

connection.Close(); // Закрываем подключение

}

}

static public void Insert(string insert) // Выполнение запроса insert, delete, update

{

// МЕТОД ВЫПОЛНЕНИЯ ЗАПРОСОВ INSERT, DELETE, UPDATE

try

{

connection.Open(); // Открываем подключение

SqlCommand command = new SqlCommand(insert, connection); // Выполняем запрос

int num = command.ExecuteNonQuery();

MessageBox.Show("Обработано " + num + " записей"); // Сообщаем результаты выполнения запроса

}

catch

{

MessageBox.Show("Ошибка подключения");

}

finally

{

connection.Close(); // Закрываем подключение

}

}

// МЕТОД ИСПРАВЛЕНИЯ ТАБЛИЦЫ dt

static public void TabUpdate(string select, ref DataTable dt)

{

try

{

connection.Open(); // Открываем подключение

SqlDataAdapter adapter = new SqlDataAdapter(select, connection);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(adapter);

adapter.Update(dt); // Обновляем таблицу

MessageBox.Show("Данные обновлены");

}

catch

{

MessageBox.Show("Ошибка подключения");

}

finally

{

connection.Close(); // закрываем подключение

}

}

static class DataID

{

public static Int64 p { get; set; }

}

}

}