

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра информационных систем**

**КУРСОВАЯ РАБОТА**  
**по дисциплине «Управление данными»**  
**Тема: Разработка базы данных и программы для завуча школы**

Студент гр. 0361

\_\_\_\_\_

Бушуев Д.И.

Преподаватель

\_\_\_\_\_

Татарникова Т.М.

Санкт-Петербург

2022

## ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Студент Бушуев Д.И.

Группа 0361

Тема работы: Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для завуча школы.

Исходные данные:

**Вариант 3.** Для каждого учителя в БД должны храниться сведения о предметах, которые он преподает, номере кабинета, который за ним закреплен, номера классов, в которых он ведет занятия, номере урока и дне, когда он ведет уроки. Существуют учителя, которые не имеют собственного кабинета. Об учениках должны храниться следующие сведения: фамилия и имя, в каком классе учится, какая оценка по каждому предмету получена. Ученик может исправить полученную оценку. Завуч может добавить информацию о новом учителе или ученике, а также удалить - выбывших.

Завучу могут потребоваться следующие сведения: какой предмет будет в заданном классе, например, во вторник на заданном уроке; кто из учителей преподает в заданном классе; в каком кабинете будет 5-й урок в среду у некоторого класса; в каких классах преподает учитель заданный предмет; расписание на заданный день недели для класса. Завуч может вносить следующие изменения: вносить информацию о новом учителе; удалять запись об ученике; изменить оценку ученику.

Необходимо предусмотреть возможность выдачи справки о количестве учеников в данном классе и отчета о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число двоечников, хорошистов и отличников).

Содержание пояснительной записки: Задание на курсовую работу, введение, анализ предметной области, обоснование модели данных, обоснование выбора

СУБД, описание функций групп пользователей, описание функций управления данными, организация защиты БД, заключение, список использованных источников, Приложение 1. Руководство пользователя БД, Приложение 2. Листинг программного кода.

Предполагаемый объем пояснительной записки:

Не менее 20 страниц.

Дата выдачи задания: 01.09.2022

Дата сдачи реферата: 23.12.2022

Дата защиты реферата: 23.12.2022

Студент

\_\_\_\_\_

Бушуев Д.И.

Преподаватель

\_\_\_\_\_

Татарникова Т.М.

## **АННОТАЦИЯ**

Задачей курсовой работы является формирование способностей: использовать основы знаний проектирования баз данных, использовать навыки программирования, к самоорганизации и самообразованию.

## **SUMMARY**

The objective of the course work is the formation of abilities: to use the basic knowledge of database design, to use programming skills, to self-organization and self-education.

## СОДЕРЖАНИЕ

	Введение	6
1.	Анализ предметной области	8
1.1	Описание объектов предметной области	8
1.2	Формулировка задач	9
1.3	Описание алгоритмов решения задач	9
1.4	Определение групп пользователей	10
1.5	Входные и выходные документы	10
2.	Обоснование модели данных	12
3.	Обоснование выбора СУБД	13
4.	Описание функций групп пользователей	14
5.	Описание функций управления данными	15
5.1	Хранение	15
5.2	Манипулирование	16
5.3	Доступ к данным	18
5.4	Предоставление запрашиваемых данных пользователю	18
6.	Организация защиты базы данных	19
6.1	Описание ограничений целостности	19
6.2	Рекомендуемые средства физической защиты	20
6.3	Описание процедуры подтверждения подлинности	20
	Заключение	21
	Список использованных источников	22
	Приложение 1. Руководство пользователя БД	23
	Приложение 2. Листинг программного кода	31

## ВВЕДЕНИЕ

Что же такое база данных? База данных – это хранилище определенной информации. Она используется во многих сферах, где необходимо собирать и сохранять большой объем материала. Например, в различные веб-разработки, интернет-магазины, сайты покупки билетов и так далее. База данных включает все, что содержит информацию о вашей организации и о ваших клиентах. Для интернет-магазинов это каталоги, прайс-листы, данные покупателей, указанные в их профиле. Все, что хранится в базе данных доступно для изменения и извлечения при необходимости.

Система управления базами данных (сокращенно СУБД) – это программное обеспечение для создания и работы с базами данных. Основной задачей СУБД является управление информацией, которая располагается как во внешней, так и в оперативной памяти. СУБД поддерживает языки баз данных, а также отвечает за копирование и восстановление информации после каких-либо сбоев.

Реляционные базы данных представляют собой базы данных, которые используются для хранения и предоставления доступа к взаимосвязанным элементам информации. Реляционные базы данных основаны на реляционной модели — интуитивно понятном, наглядном табличном способе представления данных. Каждая строка, содержащая в таблице такой базы данных, представляет собой запись с уникальным идентификатором, который называют ключом. Столбцы таблицы имеют атрибуты данных, а каждая запись обычно содержит значение для каждого атрибута, что дает возможность легко устанавливать взаимосвязь между элементами данных.

Реляционная модель подразумевает логическую структуру данных: таблицы, представления и индексы. Логическая структура отличается от физической структуры хранения. Такое разделение дает возможность администраторам управлять физической системой хранения, не меняя данных, содержащихся в логической структуре. Например, изменение имени файла базы

данных не повлияет на хранящиеся в нем таблицы. При выборе типа базы данных и продуктов на основе реляционных баз данных необходимо учитывать несколько факторов. Выбор РСУБД зависит от потребностей заказчика. Нужно задать себе следующие вопросы.

- Каковы наши требования к точности данных?
- Нужна ли нам масштабируемость? Какими объемами данных требуется управлять и каков прогнозируемый рост этих объемов?
- Насколько важно наличие параллельного доступа? Потребуется ли пользователям и приложениям одновременный доступ к данным?
- Каковы наши потребности в эффективности и надежности баз данных? Требуется ли нам высокоэффективная и надежная система? Каковы требования к скорости выполнения запросов?

В учебных заведениях преподают и учатся большое количество людей. Для того чтобы ускорить процессы нужно автоматизировать учебный процесс. Необходимо грамотно построить систему хранения данных, чтобы в дальнейшем поиск, добавление, изменение и удаление данных происходили быстро. Так же для отчетности важно настроить автоматическое составление отчетов и справок по запросу пользователя.

Как итог логичным решением является грамотно составленная база данных. Нужно провести системный анализ предметной области и определиться с видом будущего программного продукта.

## 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

### 1.1. Описание объектов предметной области

Сотрудникам учебного заведения необходимо централизованно собирать информацию об учебном процессе. В качестве будем использовать таблицы. После анализа задания были выделены следующие объекты и их атрибуты:

- Id\_teachers:
  - id\_teacher,
  - name\_1,
  - name\_2,
  - name\_3,
  - fix\_cl\_room;
- Id\_students:
  - id\_student,
  - name\_1,
  - name\_2,
  - grade;
- Id\_subjects:
  - id\_subject,
  - subject;
- Students:
  - id\_student,
  - subject,
  - mark;
- Teachers:
  - id\_teacher,
  - subject,
  - grade,
  - day,



- n\_lesson,
- room;
- Grade:
  - grade;
- N\_lesson:
  - n\_lesson;
- Day:
  - day.

## **1.2. Формулировка задач**

Задача базы данных хранить информацию об учениках и учителях, следить за успеваемостью учеников, печатать отчеты о количестве учеников и отчет о работе школы.

## **1.3. Описание алгоритмов решения задач**

Чтобы исправить оценку ученику, достаточно найти id ученика, а после посмотреть его оценки по предмету в таблице students.

Чтобы редактировать информацию об ученике, достаточно в таблице id\_students добавить новую запись, при этом добавлять в существующий класс в школе. Чтобы удалить, достаточно проверить на наличие в этой таблице и затем удалить запись.

Чтобы редактировать информацию об учителе, достаточно в таблице id\_teachers добавить новую запись, при этом закрепленного кабинета может не быть. Чтобы удалить, достаточно проверить на наличие в этой таблице и затем удалить запись.

Чтобы узнать какие предметы были у класса в определенный день, достаточно по таблице n\_lesson отобразить таблицу teachers для определенного класса и дня.

Чтобы узнать список преподавателей у определенного класса, можно посмотреть таблицу teachers при этом отобразив по id учителей их ФИО.

Чтобы узнать в каком кабинете будет урок в определенный день, достаточно посмотреть таблицу teachers по нужным параметрам.

Чтобы узнать в каких классах учитель преподает определенный предмет, достаточно в таблице teachers отобразить все записи с этим классом и учителем и вычленив повторяющиеся.

Чтобы сформировать справку о количестве учеников в классе, достаточно подсчитать количество записей учеников в нужном классе, после чего сгенерировать документ и вставить информацию в него.

Чтобы сформировать отчет о работе школы, нужно вывести результаты нескольких запросов.

Чтобы сделать резервную копию БД, нужно воспользоваться функционалом sqlite3.

#### **1.4. Определение групп пользователей**

Для решения задачи потребуется 2 группы пользователей:

- Заведующий учебной частью;
- Учитель.

Завуч будет иметь доступ ко всему функционалу программы и базы данных, а также будет иметь возможность генерировать отчет о работе школы и справки о количестве учеников в классе. В группу пользователей не были выделены ученики так, как право изменять оценки может спровоцировать соблазн редактирования своей успеваемости. Так что для того, чтобы исправить оценку ученик необходимо обратиться к учителю по предмету, который подтвердит изменение оценки действиями в системе.

#### **1.5. Входные и выходные документы**

В качестве *входных данных* для создаваемой базы используются следующая информация:

Информация о людях, задействованных в процессе обучения – учителя (ФИО, предмет, закрепленный кабинет) и ученики (ФИ, класс обучения), а также

таблицы успеваемости учеников (Оценки по каждому из предметов) и расписание уроков (номер урока, предмет и кабинет, в котором будет этот урок).

*Выходные данные, генерируемые программой:*

- Справка о количестве учеников. В ней находится информация о количестве учеников в заданном, в программе, классе;
- Отчет о работе школы. В нем находится информация об количестве учителей по предметам, количестве кабинетов в школе, количестве учеников в каждом из классов и количество учеников по категориям успеваемости.

## 2. ОБОСНОВАНИЕ МОДЕЛИ ДАННЫХ

В качестве модели данных была выбрана реляционная база данных, потому что данные о школе структурированы. Ниже приведена схема для созданной базы данных (Рисунок 1).



Рисунок 1 – Схема базы данных

### **3. ОБОСНОВАНИЕ ВЫБОРА СУБД**

В качестве СУБД была выбрана SQLite, потому что он не использует парадигмы клиент-сервер, представляет собой библиотеку, с которой программа компонуется, и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite.

Такой подход уменьшает накладные расходы, время отклика и упрощает программу. Для Завуча такой подход удобен, все отчеты он сможет составлять по мере необходимости и информацию заносить сразу в единственный стандартный файл на том компьютере, на котором работает.

Благодаря особенностям архитектуры SQLite работает быстро. Компоненты СУБД встроены в приложение и вызываются в том же процессе. Поэтому доступ к ним быстрее, чем при взаимодействии между разными процессами.

SQLite отличают малый размер, простота решений и легкость администрирования. Для преподавательского состава школ РФ, понятность и удобство использования важно.

SQLite считается надежной СУБД с минимальным риском непредсказуемого поведения.

Так же SQLite является бесплатным ПО, что является плюсом для школы на государственном обеспечении.

Недостатки SQLite, например, такие как ограниченная поддержка типов данных, отсутствие хранимых процедур в решении для завуча школы не играют существенной роли.

#### 4. ОПИСАНИЕ ФУНКЦИЙ ПОЛЬЗОВАТЕЛЕЙ

Функции пользователей проиллюстрированы на рисунке 2 ниже. В пользователи не включены ученики, т. к. существует соблазн исправления оценок на «отлично». Проблема решается контролем исправления оценок учителем.



Рисунок 2 – Функции пользователей

Описание прав доступа приведено в таблице ниже.

Таблица 1 - Права доступа

<b>Объект</b>	<b>Завуч</b>	<b>Учитель</b>
День	SIUD	S
Класс	SIUD	S
Номер урока	SID	S
ID Предметов	SIUD	S
ID Учеников	SIUD	S
ID Учителей	SIUD	S
Ученики	SID	SIUD
Учителя	SIUD	S

## 5. ОПИСАНИЕ ФУНКЦИЙ УПРАВЛЕНИЯ ДАННЫМИ

### 5.1. Хранение

Создание таблиц необходимых для создания полной базы данных происходит в программе с помощью следующих SQL запросов:

```
CREATE TABLE IF NOT EXISTS n_lesson(  
    n_lesson INTEGER  
);  
CREATE TABLE IF NOT EXISTS day(  
    day TEXT  
);  
CREATE TABLE IF NOT EXISTS grade(  
    grade TEXT  
);  
CREATE TABLE IF NOT EXISTS id_subjects(  
    id_subject INTEGER NOT NULL,  
    subject TEXT,  
    PRIMARY KEY("id_subject" AUTOINCREMENT)  
);  
CREATE TABLE IF NOT EXISTS id_students(  
    id_student INTEGER NOT NULL,  
    name_1 TEXT,  
    name_2 TEXT,  
    grade TEXT,  
    PRIMARY KEY("id_student" AUTOINCREMENT)  
);  
CREATE TABLE IF NOT EXISTS id_teachers(  
    id_teacher INTEGER NOT NULL,  
    name_1 TEXT,  
    name_2 TEXT,  
    name_3 TEXT,  
    fix_cl_room TEXT,  
    PRIMARY KEY("id_teacher" AUTOINCREMENT)  
);  
CREATE TABLE IF NOT EXISTS students(  
    id_student INTEGER,  
    subject TEXT,  
    mark INTEGER  
);  
CREATE TABLE IF NOT EXISTS teachers(  
    id_teacher INTEGER,  
    subject TEXT,  
    grade TEXT,  
    day TEXT,  
    n_lesson INTEGER,  
    room TEXT  
);
```

## 5.2. Манипулирование

Оценка ученика изменяется с помощью следующих SQL запросов:

```
SELECT name_2, subject, mark FROM students
JOIN id_students ON id_students.id_student = students.id_student
WHERE students.id_student = (SELECT id_student FROM id_students
WHERE name_1 = ? AND name_2 = ?);
```

```
SELECT name_2, subject, mark as 'оценка' FROM students
JOIN id_students ON id_students.id_student = students.id_student
WHERE students.id_student = (SELECT id_student FROM id_students
WHERE name_1 = ? AND name_2 = ?) AND subject = ?;
```

```
UPDATE students SET mark = ?
WHERE (students.id_student = (SELECT id_student FROM id_students
WHERE name_1 = ? AND name_2 = ?))
AND (rowid = (SELECT ROWID FROM students WHERE mark = ?
AND (students.id_student = (SELECT id_student FROM id_students
WHERE name_1 = ? AND name_2 = ?)) AND (subject = ?))));
```

Редактирование информации об ученике происходит с помощью следующих SQL запросов:

```
SELECT grade FROM grade WHERE grade = ?;
INSERT INTO id_students (name_1, name_2, grade) VALUES (?, ?, ?);
```

```
SELECT grade FROM grade WHERE grade = ?
DELETE FROM id_students WHERE name_1 = ? AND name_2 = ? AND grade = ?
AND rowid = (SELECT ROWID FROM id_students
WHERE name_1 = ? AND name_2 = ? AND grade = ?);
```

```
JOIN id_students ON id_students.id_student = students.id_student
WHERE students.id_student = (SELECT id_student FROM id_students
WHERE name_1 = ? AND name_2 = ?);
```

Редактирование информации об учителе происходит с помощью следующих SQL запросов:

```
INSERT INTO id_teachers (name_1, name_2, name_3, fix_cl_room)
VALUES (?, ?, ?, ?);
```

```
INSERT INTO id_teachers (name_1, name_2, name_3) VALUES (?, ?, ?);
```

```
DELETE FROM id_teachers
WHERE name_1 = ? AND name_2 = ? AND name_3 = ?
AND rowid = (SELECT ROWID FROM id_teachers
WHERE name_1 = ? AND name_2 = ? AND name_3 = ?)
```



Чтобы узнать какие предметы будут в заданном классе в заданный день будут использоваться следующие SQL запросы:

```
SELECT n_lesson, subject, room, name_1, name_2, name_3
FROM teachers
JOIN id_teachers ON id_teachers.id_teacher = teachers.id_teacher
WHERE grade = ? AND day = ?
```

Чтобы узнать кто из учителей преподает в заданном классе будут использоваться следующие SQL запросы:

```
SELECT DISTINCT subject, name_1, name_2, name_3
FROM teachers
JOIN id_teachers ON id_teachers.id_teacher = teachers.id_teacher
WHERE grade = ?
```

Чтобы узнать в каком кабинете будет заданный урок в заданный день у заданного класса будут использоваться следующие SQL запросы:

```
SELECT room FROM teachers
WHERE (day = ? AND grade = ?) AND n_lesson = ?;
```

Чтобы узнать в каких классах преподает определенный учитель заданный предмет будут использоваться следующие SQL запросы:

```
SELECT DISTINCT grade FROM teachers
JOIN id_teachers ON id_teachers.id_teacher = teachers.id_teacher
WHERE (name_1 = ? AND name_2 = ? AND name_3 = ?) AND subject = ?
ORDER BY grade
```

Чтобы сформировать справку о количестве учеников в классе будут использоваться следующие SQL запросы:

```
SELECT count(id_student) as "Количество учеников"
FROM id_students WHERE grade = ?
```

Чтобы сформировать отчет о работе школы будут использоваться следующие SQL запросы:

```
SELECT subject, count(DISTINCT id_teacher)
FROM teachers group by subject;
```

```
SELECT count(DISTINCT room) FROM teachers WHERE not (room = "");
```

```
SELECT grade, count(DISTINCT id_teacher)
FROM teachers GROUP BY grade;
```

```
SELECT count(DISTINCT id_student) FROM students
```

Вместо знаков вопроса во всех запросах используются введенные с клавиатуры данные, которые вводятся в приложении.

### **5.3. Доступ к данным**

Чтобы разграничить доступ к данным в SQLite нет встроенной системы для этого. Так что права доступа будут обозначены на уровне файлов в системе. Для того чтобы грамотно разграничить права пользователей и исключить несанкционированный доступ необходимо выставить разрешения на чтение и запись.

### **5.4. Предоставление запрашиваемых данных пользователю**

Программа генерирует справку о количестве учеников в классе и отчет о работе школы.

Для генерации справки информация получается с помощью SQL запросов:

```
SELECT count(id_student) as "Количество учеников"  
FROM id_students WHERE grade = ?
```

После получения информации все записывается в файл .docx.

Для генерации отчета о работе школы используются SQL запросы:

```
SELECT subject, count(DISTINCT id_teacher)  
FROM teachers group by subject;
```

```
SELECT count(DISTINCT room) FROM teachers WHERE not (room = "");
```

```
SELECT grade, count(DISTINCT id_teacher)  
FROM teachers GROUP BY grade;
```

```
SELECT count(DISTINCT id_student) FROM students
```

Результат так же записывается в файл с разрешением .docx.

## 6. ОРГАНИЗАЦИЯ ЗАЩИТЫ БД

### 6.1. Описание ограниченной целостности

В таблице представлены ограничения целостности для объектов.

Объект	Минимальный размер	Максимальный размер	Ограничение «NOT NULL»	Ограничение уникальности
<b>id_teachers</b>				
id	-	-	+	+
name_1	-	-	+	-
name_2	-	-	+	-
name_3	-	-	-	-
fix_cl_room		-	+	-
<b>id_students</b>				
id_student	-	-	+	+
name_1	-	-	+	-
name_2	-	-	+	-
grade	-	-	+	-
<b>id_subjects</b>				
id_subject	-	-	+	+
subject	-	-	+	-
<b>day</b>				
day	-	-	+	-
<b>grade</b>				
grade	-	-	+	-
<b>n_lesson</b>				
n_lesson	-	-	+	-
<b>teachers</b>				
id_teacher	-	-	+	+
subject	-	-	+	-
grade	-	-	+	-
day	-	-	+	-
n_lesson	-	-	+	-
room	-	-	-	-
<b>students</b>				
id_student	-	-	+	+
subject	-	-	+	-
mark	2	5	+	-

## **6.2. Рекомендуемые средства физической защиты**

В программе предусмотрена возможность создания резервной копии, рекомендуется делать ее каждую неделю во избежание существенных потерь данных.

## **6.3. Описание процедуры подтверждения подлинности**

Подтверждение личности происходит с помощью выбора пользователя при запуске программы в формате диалога. Так как программой пользуются только доверенные пользователи: учителя и завуч, достаточно ограничить доступ к программе и базе данных запретив доступ к помещению или установив пароль на компьютер.

## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была создана программа для управления базой данных для завуча школы в режиме диалога. Для написания программы были использованы язык Python и СУБД SQLite 3.

Программа отвечает всем требованиям заказчика и работает в удобном режиме диалога. База данных позволяет централизовать информацию учебного заведения и с помощью программы эффективно организовать работу государственного учреждения.

Программа предоставляет следующие функции:

- Добавление, удаление, изменение информации об учеников;
- Создания расписания;
- Редактирование успеваемости учеников;
- Генерирование справок о количестве учеников в конкретном классе;
- Генерирование отчета о работе школы.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Amazon Web Services, Inc. или дочерние организации, 2022. URL: <https://aws.amazon.com/ru/relational-database/> (дата обращения 15.12.2022);
2. ООО «ГикБрейнс» URL: <https://gb.ru/blog/что-такое-база-dannykh/> (дата обращения 15.12.2022);
3. 2022 Oracle URL: <https://www.oracle.com/cis/database/what-is-a-relational-database/> (дата обращения 15.12.2022);
4. PyQt 5 Documentation URL: <https://doc.qt.io/qtforpython/> (дата обращения 15.11.2022);
5. SQLite Documentation URL: <https://www.sqlite.org/docs.html> (дата обращения 15.11.2022)

## ПРИЛОЖЕНИЕ А. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

### 1. Запуск и выбор пользователя

Запуск программы встречает нас диалоговым окном выбора пользователей, в качестве примера работы программы зайдём под заведующим учебной работой у него гораздо больше возможностей нежели у учителя. При некорректном вводе данных программа корректно обрабатывает ошибки.

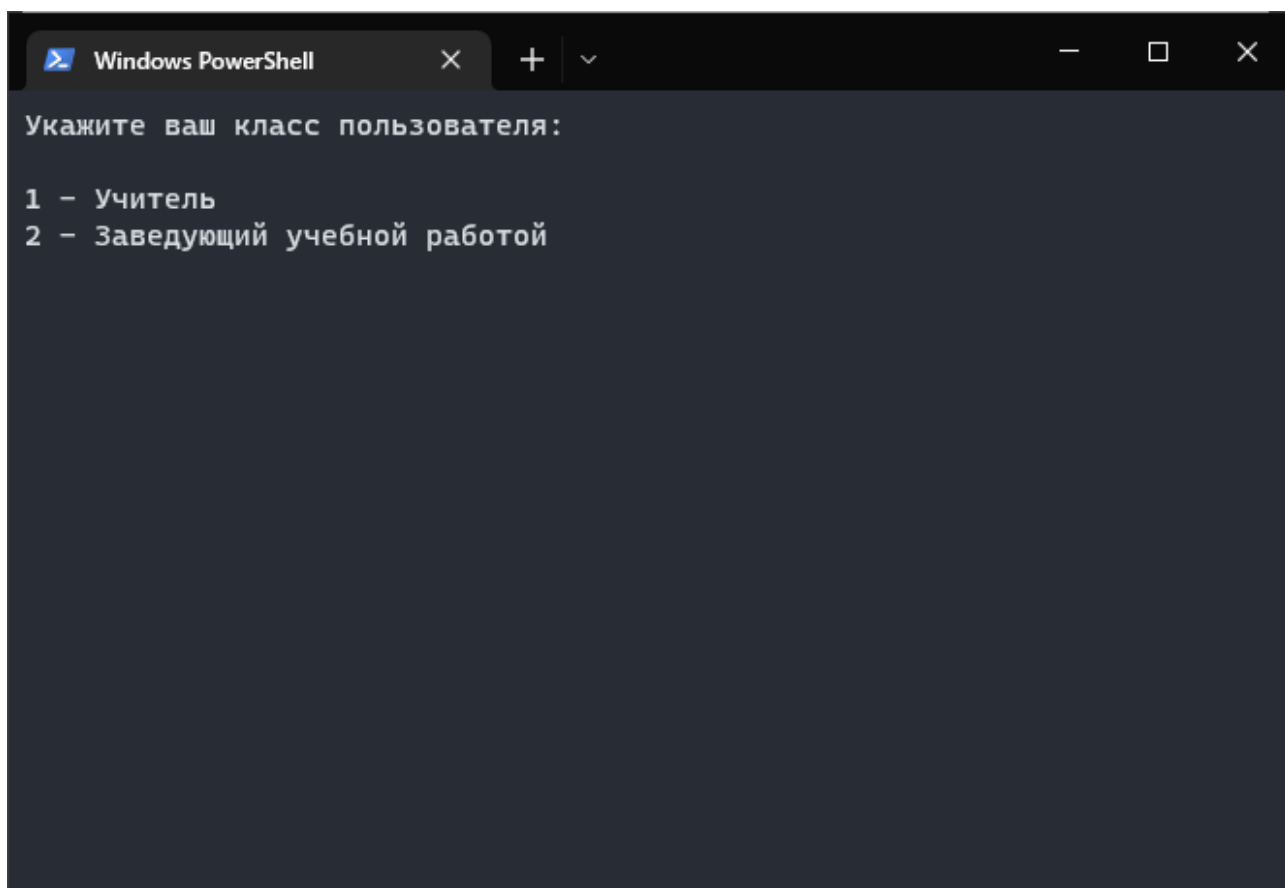


Рисунок 3 – Окно при запуске программы

Далее мы видим окно выбора возможностей (Рисунок 4), разберем каждый пункт в подробностях.

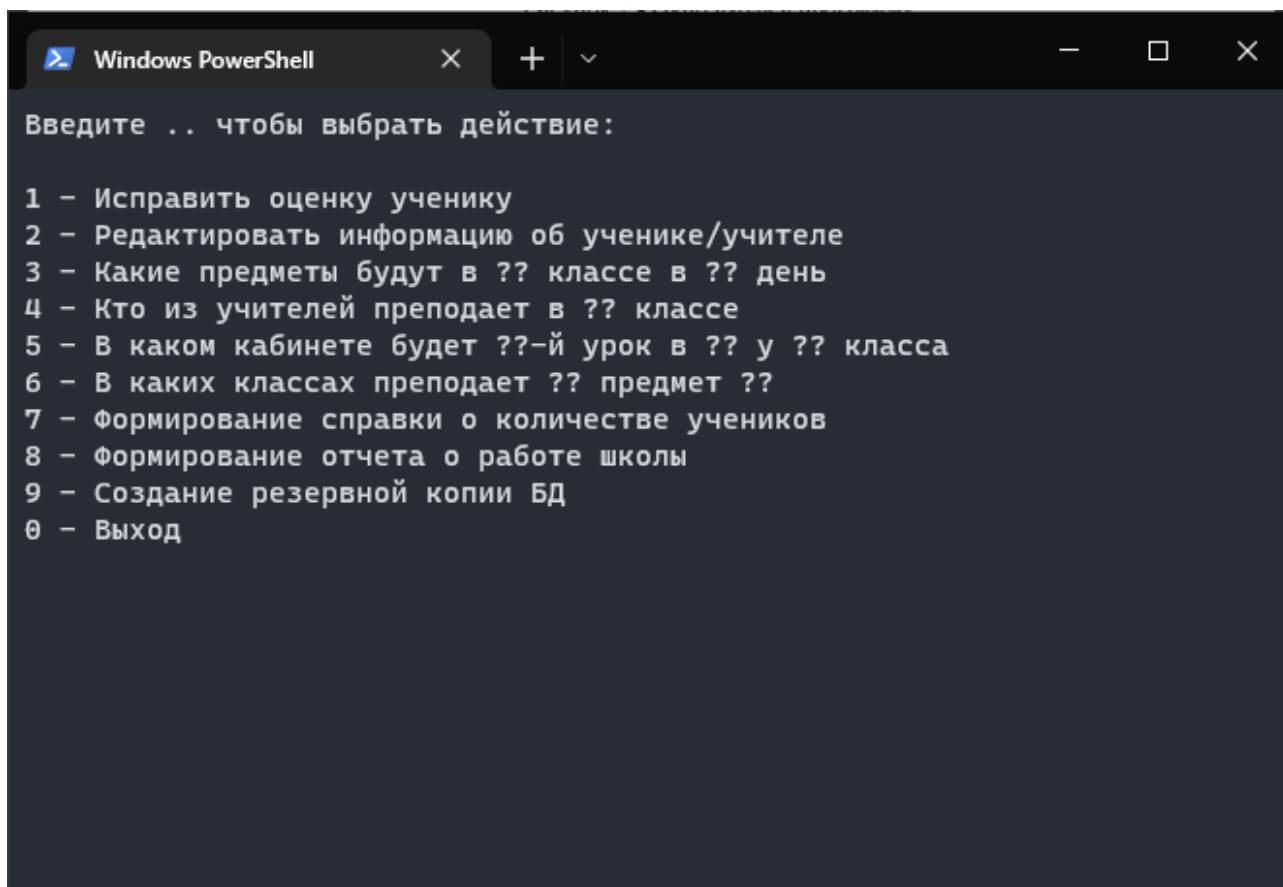


Рисунок 4 – Окно выбора возможностей



## 2. Исправление оценки ученику

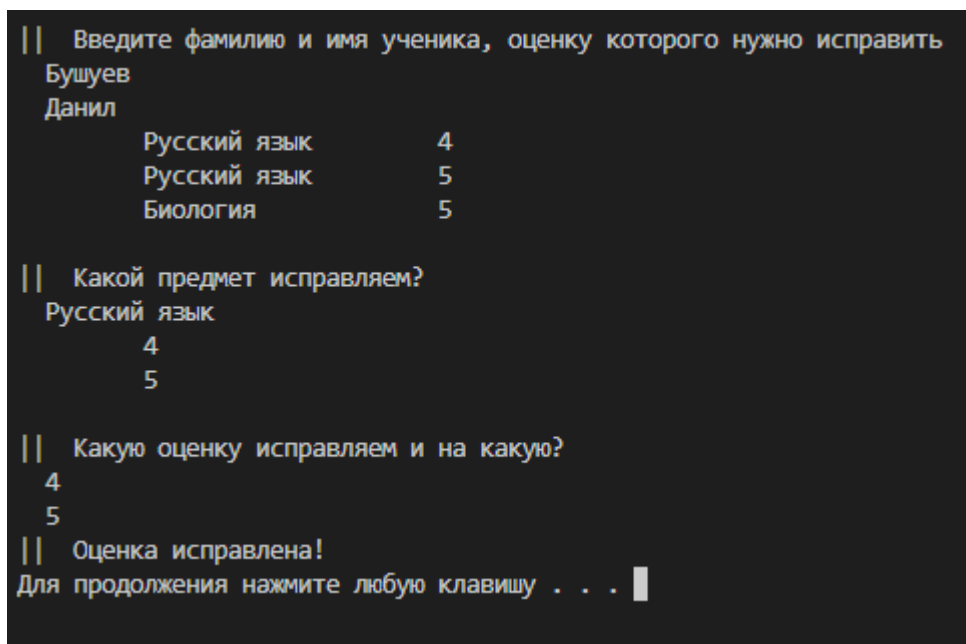


Рисунок 5 – Окно первого выбора

При выборе первой возможности программы, от пользователя требуется ввести фамилию и имя оценки которого нужно исправить. А затем нужно ввести предмет, который исправляем. А после нужно ввести оценку, которую исправляем и оценку, на которую исправляем. При некорректном вводе данных программа корректно обрабатывает ошибки.

## 3. Редактирование информации об ученике/учителе

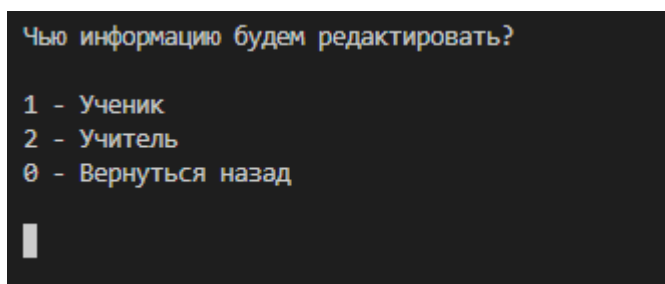


Рисунок 6 – Выбор что редактировать

Далее программа просит от пользователя выбрать, что делать с информацией: добавлять или удалять ее (Рисунок 7). Диалоговое окно одинаково возникает при выборе на предыдущем шаге.

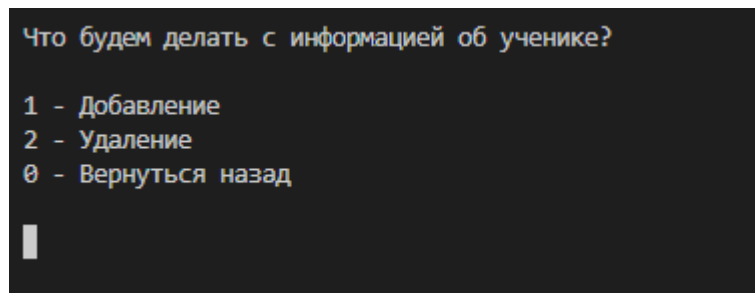


Рисунок 7 – Выбор что делать с информацией

На следующих шагах программа при добавлении или удалении просит ввести данные ученика (Рисунок 8). При добавлении или удалении информации об учителе программа просит ввести данные об учителе (Рисунок 9).

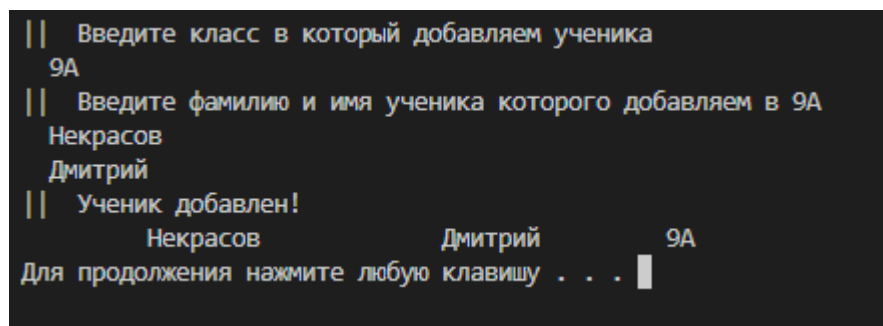


Рисунок 8 – Добавление ученика

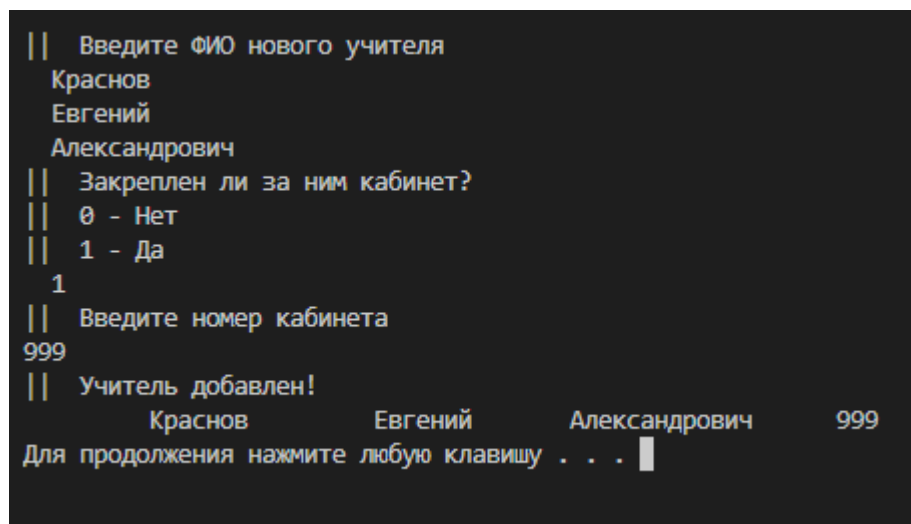


Рисунок 9 – Добавление учителя

```

|| Введите класс из которого удаляем ученика
9А
|| Введите фамилию и имя ученика в 9А которого удаляем
Некрасов
Дмитрий
|| Ученик удален
Для продолжения нажмите любую клавишу . . . █

```

Рисунок 10 – Удаление ученика

```

|| Введите ФИО учителя
Краснов
Евгений
Александрович
|| Учитель удален
Для продолжения нажмите любую клавишу . . . █

```

Рисунок 11 – Удаление учителя

Удаление происходит по такому же принципу, что и добавление – в формате диалога с пользователем.

#### 4. Вывод расписания на конкретный день для заданного класса

В формате диалога пользователь вводит нужные ему параметры и получает на экране таблицу с расписанием уроков, где первый столбец это номер урока, второй столбец это предмет, третий столбец – кабинет и в конце ФИО учителя (Рисунок 12).

```

|| В какой день(ДД.ММ.ГГГГ) будет урок?
12.12.2022
|| У какого класса?
7А

```

1	География	211	Геообжов	Михаил	Игоревич
2	Английский язык	218	Англова	Василиса	Степановна
3	Физика	208	Физикова	Любовь	Гордеевна
4	Русский язык	104	Руслитова	Зоя	Михайловна
5	Литература	104	Руслитова	Зоя	Михайловна
6	Информатика	215	Инфотехов	Кирилл	Леонидович

```

Для продолжения нажмите любую клавишу . . . █

```

Рисунок 12 – Выбор третьего варианта в списке возможностей

## 5. Вывод списка учителей для класса

Работа все так же, как в 4 пункте происходит в формате диалога (Рисунок 13).

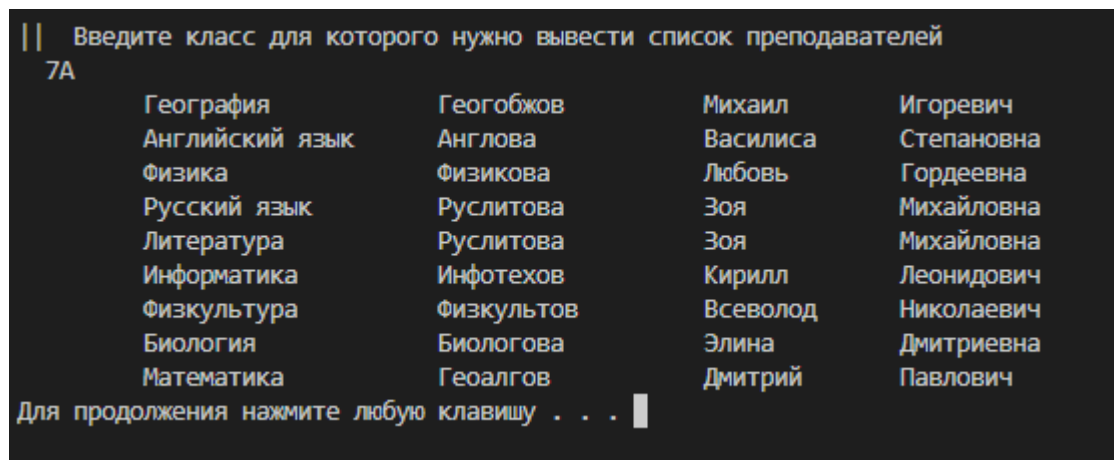


Рисунок 13 – Список учителей для класса

## 6. Поиск кабинета

В формате диалога пользователь вводит данные и программа находит для него нужный кабинет (Рисунок 14).

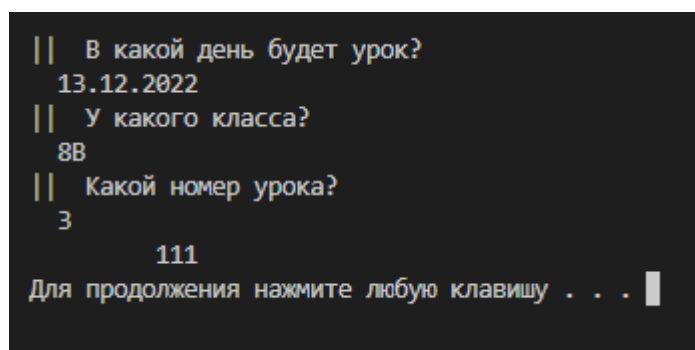


Рисунок 14 – Поиск кабинета

## 7. Получение информации о классах, в которых преподает заданный учитель заданный предмет

```

|| Введите ФИО преподавателя
Химов
Артём
Владиславович
|| Введите название предмета
Химия
|| Артём Владиславович преподает в:
8А
8Б
8В
Для продолжения нажмите любую клавишу . . .

```

Рисунок 15 – Демонстрация работы шестой возможности программы

## 8. Формирование справки о количестве учеников

При выборе этого варианта в программе, от пользователя требуется ввести номер класса в школе, для того чтобы сгенерировать справку (Рисунок 16). Документ с информацией создастся в папке с программой (Рисунок 17).

```

|| Формирование справки о количестве учеников в классе
|| Введите класс
7А
11
|| Количество учеников в 7А
Для продолжения нажмите любую клавишу . . .

```

Рисунок 16 – Демонстрация работы создания справки

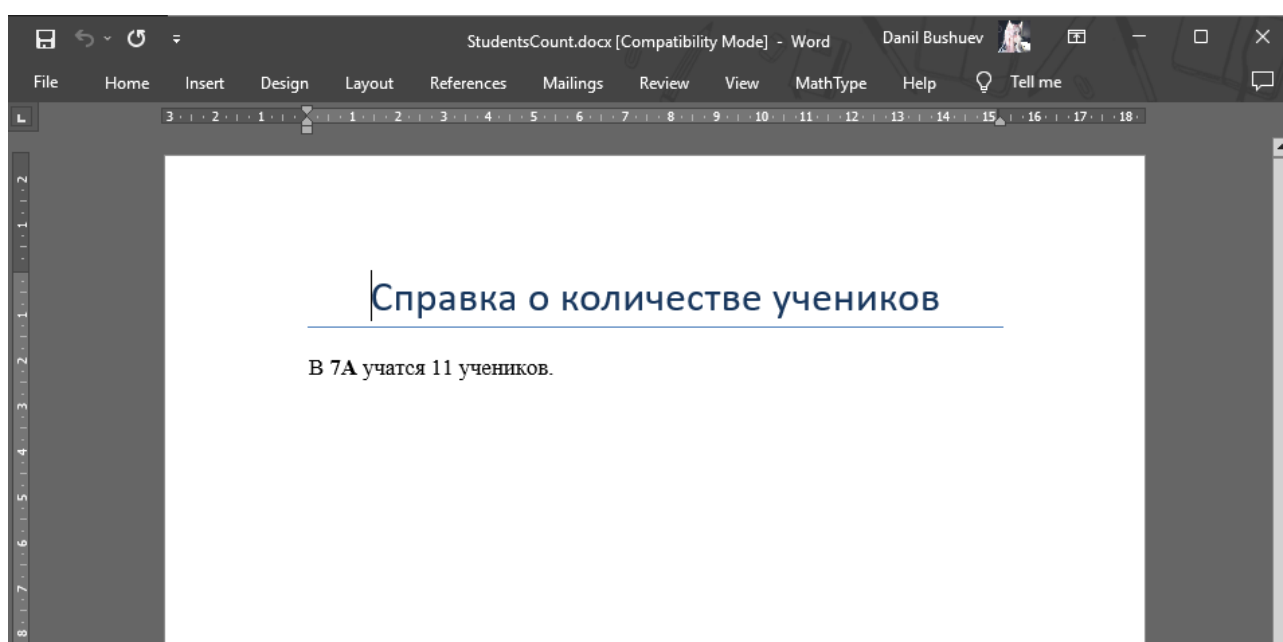


Рисунок 17 – Сгенерированный документ 1

## 9. Формирование отчета о работе школы

Сразу после выбора варианта в окне приложения (Рисунок 18), в папке с программой появится отчет с нужной информацией (Рисунок 19).

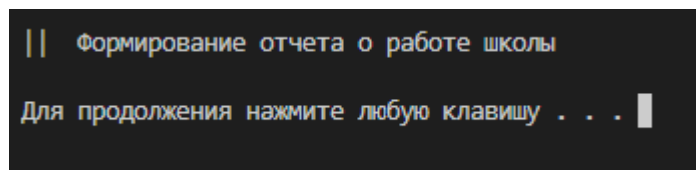
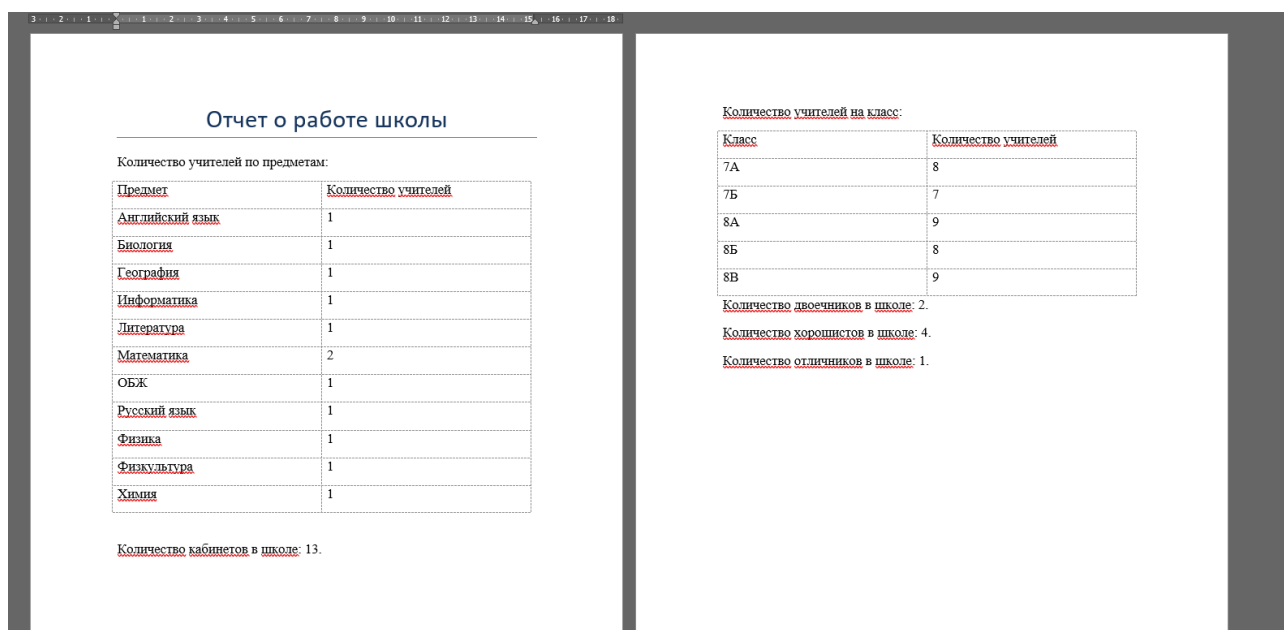


Рисунок 18 – Выполнение действия программой



Количество учителей по предметам:	
Предмет	Количество учителей
Английский язык	1
Биология	1
География	1
Информатика	1
Литература	1
Математика	2
ОБЖ	1
Русский язык	1
Физика	1
Физкультура	1
Химия	1
Количество кабинетов в школе: 13.	

Класс	Количество учителей
7А	8
7Б	7
8А	9
8Б	8
8В	9
Количество двоечников в школе: 2.	
Количество хорошистов в школе: 4.	
Количество отличников в школе: 1.	

Рисунок 19 – Сгенерированный отчет о работе школы

## 10. Возможные проблемы и их решения

При неправильном вводе данных в базу нужно удалить эти данные с помощью функционала приложения или с помощью приложения для визуализации СУБД, например, db browser. Так же возможна потеря файла с базами данных, в таком случае необходимо восстановить базу данных из резервной копии

## ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММНОГО КОДА

```
# -*- coding: utf-8 -*-
import sqlite3 as sq
import os
from docx import Document
from docx.shared import Pt
from docx.enum.text import WD_PARAGRAPH_ALIGNMENT

def sq_init():
    with sq.connect("school.db") as db:
        cur = db.cursor()
        #Создание пустой базы данных
        query = """
        CREATE TABLE IF NOT EXISTS n_lesson(
            n_lesson INTEGER
        );
        CREATE TABLE IF NOT EXISTS day(
            day TEXT
        );
        CREATE TABLE IF NOT EXISTS grade(
            grade TEXT
        );
        CREATE TABLE IF NOT EXISTS id_subjects(
            id_subject INTEGER NOT NULL,
            subject TEXT,
            PRIMARY KEY("id_subject" AUTOINCREMENT)
        );
        CREATE TABLE IF NOT EXISTS id_students(
            id_student INTEGER NOT NULL,
            name_1 TEXT,
            name_2 TEXT,
            grade TEXT,
            PRIMARY KEY("id_student" AUTOINCREMENT)
        );
        CREATE TABLE IF NOT EXISTS id_teachers(
            id_teacher INTEGER NOT NULL,
            name_1 TEXT,
            name_2 TEXT,
            name_3 TEXT,
            fix_cl_room TEXT,
            PRIMARY KEY("id_teacher" AUTOINCREMENT)
        );
        CREATE TABLE IF NOT EXISTS students(
            id_student INTEGER,
            subject TEXT,
            mark INTEGER
        );
        CREATE TABLE IF NOT EXISTS teachers(
            id_teacher INTEGER,
            subject TEXT,
            grade TEXT,
            day TEXT,
            n_lesson INTEGER,
            room TEXT
        )
        """
```

```

        cur.executescript(query)
def sq_dump_w():
    with sq.connect("school.db") as db:
        cur = db.cursor()
        with open("sql_dump.sql", "w") as f:
            for sql in db.iterdump():
                f.write(sql)
def sq_dump_r():
    with sq.connect("school.db") as db:
        cur = db.cursor()
        with open("sql_dump.sql", "r") as f:
            sql = f.read()
            cur.executescript(sql)

def choose_user():
    os.system('cls')
    print("Укажите ваш класс пользователя:\n")
    print("1 - Учитель")
    print("2 - Заведующий учебной работой")
def choose_main(user):
    os.system('cls')
    print("Введите .. чтобы выбрать действие:\n")
    print("1 - Исправить оценку ученику")
    if user == '2':
        print("2 - Редактировать информацию об ученике/учителе")
        print("3 - Какие предметы будут в ?? классе в ?? день")
        if user == '2':
            print("4 - Кто из учителей преподает в ?? классе")
            print("5 - В каком кабинете будет ??-й урок в ?? у ?? класса")
            if user == '2':
                print("6 - В каких классах преподает ?? предмет ??")
                print("7 - Формирование справки о количестве учеников")
                print("8 - Формирование отчета о работе школы")
                print("9 - Создание резервной копии БД")
            print("0 - Выход\n")
def choose_key2_1():
    os.system('cls')
    print("Чью информацию будем редактировать?\n")
    print("1 - Ученик") #49
    print("2 - Учитель") #50
    print("0 - Вернуться назад\n") #48
def choose_key2_2(key):
    os.system('cls')
    if key == 1:
        print("Что будем делать с информацией об ученике?\n")
    if key == 2:
        print("Что будем делать с информацией об учителе?\n")
    print("1 - Добавление") #49
    print("2 - Удаление") #50
    print("0 - Вернуться назад\n") #48

def key1():
    try:
        db = sq.connect("school.db")
        cur = db.cursor()
        cur.execute("""Begin;""")
        print("|| Введите фамилию и имя ученика, оценку которого нужно исправить")
        name_1 = input(" ")
        name_2 = input(" ")
        request = ""

```



```

        SELECT name_2, subject, mark
        FROM students
        JOIN id_students ON id_students.id_student = students.id_student
        WHERE students.id_student =
        (SELECT id_student FROM id_students
        WHERE name_1 = ? AND name_2 = ?);
    """
    cur.execute(request, (name_1, name_2))
    output = cur.fetchall()
    #print (output)
    if output == []:
        print("|| Такого ученика нет")
    else:
        for row in output:
            print ("\t", '%-20s %-15d' % (row[1], row[2]))
        print("\n|| Какой предмет исправляем?")
        subject = input(" ")
        request = """
            SELECT name_2 as 'ученик', subject as 'предмет', mark as 'оценка'
            FROM students
            JOIN id_students ON id_students.id_student = students.id_student
            WHERE students.id_student =
            (SELECT id_student FROM id_students
            WHERE name_1 = ? AND name_2 = ?)
            AND subject = ?;
        """
        cur.execute(request, (name_1, name_2, subject))
        output = cur.fetchall()
        #print (output)
        if output == []:
            print("|| Некорректный ввод или оценок нет, исправлять нечего")
        else:
            for row in output:
                print ("\t", '%-20s' % row[2])
            print("\n|| Какую оценку исправляем и на какую?")
            mark_to = input(" ")
            mark = input(" ")
            request = """
                UPDATE students SET mark = ?
                WHERE (students.id_student = (SELECT id_student FROM id_students
                WHERE name_1 = ? AND name_2 = ?))
                AND (rowid = (SELECT ROWID FROM students
                WHERE mark = ? AND (students.id_student = (SELECT id_student FROM id_students
                WHERE name_1 = ? AND name_2 = ?)) AND (subject = ?)));
            """
            cur.execute(request, (mark, name_1, name_2, mark_to, name_1, name_2, subject))
            db.commit()
            print("|| Оценка исправлена!")
    os.system('pause')

except sq.Error as e:
    if db: db.rollback()
    print("|| Ошибка выполнения запроса ", e)
    os.system('pause')
finally:
    if db: db.close()

def key2():
    choose_key2_1()
    key = input()
    while key != '0':

```

```

choose_key2_1()
if key == '1':
    os.system('cls')
    while key != '0':#Ученик
        choose_key2_2(1)
        key = input()
        if key == '1':#Добавление
            os.system('cls')
            try:
                db = sq.connect("school.db")
                cur = db.cursor()
                cur.execute("""Begin;""")
                print("|| Введите класс в который добавляем ученика")
                grade = input(" ")
                request = """
                SELECT grade FROM grade WHERE grade = ?;
                """
                cur.execute(request, (grade,))
                output = cur.fetchall()
                #print (output)
                if output == []:
                    print("|| Такого класса в школе нет")
                else:
                    print("|| Введите фамилию и имя ученика которого добавляем в",
grade)

                    name_1 = input(" ")
                    name_2 = input(" ")
                    request = "INSERT INTO id_students (name_1, name_2, grade) VALUES (?,
?, ?);"

                    cur.execute(request, (name_1, name_2, grade,))
                    print("|| Ученик добавлен!")
                    request = "SELECT * FROM id_students WHERE name_1 = ? AND name_2 = ?
AND grade = ?;"

                    cur.execute(request, (name_1, name_2, grade,))
                    output = cur.fetchall()
                    #print (output)
                    for row in output:
                        print ("\t", '%-20s %-15s %-15s' % (row[1],row[2],row[3]))
                    db.commit()
                os.system('pause')
            except sq.Error as e:
                if db: db.rollback()
                print("|| Ошибка выполнения запроса ", e)
                os.system('pause')
            finally:
                if db: db.close()
        if key == '2':#Удаление
            os.system('cls')
            try:
                db = sq.connect("school.db")
                cur = db.cursor()
                cur.execute("""Begin;""")
                print("|| Введите класс из которого удаляем ученика")
                grade = input(" ")
                request = """
                SELECT grade FROM grade WHERE grade = ?;
                """
                cur.execute(request, (grade,))
                output = cur.fetchall()
                #print (output)

```

```

        if output == []:
            print("||  Такого класса в школе нет")
        else:
            print("||  Введите фамилию и имя ученика в", grade, "которого уда-
ляем")

            name_1 = input(" ")
            name_2 = input(" ")
            request = """
                DELETE FROM id_students WHERE name_1 = ? AND name_2 = ? AND grade
= ?
                AND rowid = (SELECT ROWID FROM id_students WHERE name_1 = ? AND
name_2 = ? AND grade = ?);
            """
            cur.execute(request, (name_1, name_2, grade, name_1, name_2, grade,))
            print("||  Ученик удален")
            db.commit()
            os.system('pause')
        except sq.Error as e:
            if db: db.rollback()
            print("||  Ошибка выполнения запроса ", e)
            os.system('pause')
        finally:
            if db: db.close()
    choose_key2_1()
    key = input()
    if key == '2':
        os.system('cls')
        while key != '0':#Учитель
            choose_key2_2(2)
            key = input()
            if key == '1':#Добавление
                os.system('cls')
                try:
                    db = sq.connect("school.db")
                    cur = db.cursor()
                    cur.execute("""Begin;""")
                    print("||  Введите ФИО нового учителя")
                    name_1 = input(" ")
                    name_2 = input(" ")
                    name_3 = input(" ")
                    print("||  Закреплен ли за ним кабинет?\n||  0 - Нет\n||  1 - Да")
                    _if = input(" ")
                    if _if == '1':
                        print("||  Введите номер кабинета")
                        fix_cl_room = input()
                        request = "INSERT INTO id_teachers (name_1, name_2, name_3,
fix_cl_room) VALUES (?, ?, ?, ?);"
                        cur.execute(request, (name_1, name_2, name_3, fix_cl_room,))
                        print("||  Учитель добавлен!")
                        request = "SELECT * FROM id_teachers WHERE name_1 = ? AND name_2 = ?
AND name_3 = ? AND fix_cl_room = ?;"
                        cur.execute(request, (name_1, name_2, name_3, fix_cl_room,))
                        output = cur.fetchall()
                        #print (output)
                        for row in output:
                            print ("\t", '%-15s %-13s %-18s %-10s' %
(row[1],row[2],row[3],row[4]))
                    if _if == '0':
                        request = "INSERT INTO id_teachers (name_1, name_2, name_3) VALUES
(?, ?, ?);"

```

```

        cur.execute(request, (name_1, name_2, name_3,))
        print("||  Учитель добавлен!")
        request = "SELECT * FROM id_teachers WHERE name_1 = ? AND name_2 = ?
AND name_3 = ?;"

        cur.execute(request, (name_1, name_2, name_3,))
        output = cur.fetchall()
        #print (output)
        for row in output:
            print ("\t", '%-15s %-13s %-18s' % (row[1],row[2],row[3]))
        db.commit()
        os.system('pause')
    except sq.Error as e:
        if db: db.rollback()
        print("||  Ошибка выполнения запроса ", e)
        os.system('pause')
    finally:
        if db: db.close()
        choose_key2_2(2)
        key = input()
    if key == '2':#Удаление
        os.system('cls')
        try:
            db = sq.connect("school.db")
            cur = db.cursor()
            cur.execute("""Begin;""")
            print("||  Введите ФИО учителя")
            name_1 = input(" ")
            name_2 = input(" ")
            name_3 = input(" ")
            request = """
            DELETE FROM id_teachers WHERE name_1 = ? AND name_2 = ? AND name_3 =
?
            AND rowid = (SELECT ROWID FROM id_teachers WHERE name_1 = ? AND
name_2 = ? AND name_3 = ?);
            """
            cur.execute(request, (name_1, name_2, name_3, name_1, name_2, name_3,))
            print("||  Учитель удален")
            db.commit()
            os.system('pause')
        except sq.Error as e:
            if db: db.rollback()
            print("||  Ошибка выполнения запроса ", e)
            os.system('pause')
        finally:
            if db: db.close()
            choose_key2_2(2)
            key = input()
            choose_key2_1()
            key = input()
    return 0
def key3():
    try:
        db = sq.connect("school.db")
        cur = db.cursor()
        cur.execute("""Begin;""")
        print("||  В какой день(ДД.ММ.ГГГГ) будет урок?")
        day = input(" ")
        print("||  У какого класса?")
        grade = input(" ")
        request = """

```

```

        SELECT n_lesson, subject, room, name_1, name_2, name_3
        FROM teachers
        JOIN id_teachers ON id_teachers.id_teacher = teachers.id_teacher
        WHERE grade = ? AND day = ?
    """
    cur.execute(request, (grade, day,))
    #print (output)
    output = cur.fetchall()
    if output == []:
        print("|| Такого расписания нет")
    else:
        for row in output:
            print ("\t", '%-3s %-20s %-4s %-18s %-13s %-20s' %
(row[0],row[1],row[2],row[3],row[4],row[5]))
            os.system('pause')
    except sq.Error as e:
        if db: db.rollback()
        print("|| Ошибка выполнения запроса ", e)
        os.system('pause')
    finally:
        if db: db.close()
def key4():
    try:
        db = sq.connect("school.db")
        cur = db.cursor()
        cur.execute("""Begin;""")
        print("|| Введите класс для которого нужно вывести список преподавателей")
        grade = input(" ")
        request = """
            SELECT DISTINCT subject, name_1, name_2, name_3
            FROM teachers
            JOIN id_teachers ON id_teachers.id_teacher = teachers.id_teacher
            WHERE grade = ?
        """
        cur.execute(request, (grade,))
        #print (output)
        output = cur.fetchall()
        if output == []:
            print("|| Преподаватели для этого класса еще не назначены")
        else:
            for row in output:
                print ("\t", '%-20s %-18s %-13s %-20s' % (row[0],row[1],row[2],row[3]))
                os.system('pause')
    except sq.Error as e:
        if db: db.rollback()
        print("|| Ошибка выполнения запроса ", e)
        os.system('pause')
    finally:
        if db: db.close()
def key5():
    try:
        db = sq.connect("school.db")
        cur = db.cursor()
        cur.execute("""Begin;""")
        print("|| В какой день будет урок?")
        day = input(" ")
        print("|| У какого класса?")
        grade = input(" ")
        print("|| Какой номер урока?")
        n_lesson = int(input(" "))

```

```

request = """
    SELECT room FROM teachers
    WHERE (day = ? AND grade = ?) AND n_lesson = ?;
"""

cur.execute(request, (day, grade, n_lesson))
#print (output)
output = cur.fetchall()
if output == []:
    print("|| Информация не найдена")
else:
    for row in output:
        print ("\t", '%-10s' % (row[0]))
    os.system('pause')
except sq.Error as e:
    if db: db.rollback()
    print("|| Ошибка выполнения запроса ", e)
    os.system('pause')
finally:
    if db: db.close()

def key6():
    try:
        db = sq.connect("school.db")
        cur = db.cursor()
        cur.execute("""Begin;""")
        print ("|| Введите ФИО преподавателя")
        name_1 = input(" ")
        name_2 = input(" ")
        name_3 = input(" ")
        print ("|| Введите название предмета")
        subject = input(" ")
        request = """
            SELECT DISTINCT grade
            FROM teachers
            JOIN id_teachers ON id_teachers.id_teacher = teachers.id_teacher
            WHERE (name_1 = ? AND name_2 = ? AND name_3 = ?) AND subject = ?
            ORDER BY grade
        """
        cur.execute(request, (name_1, name_2, name_3, subject))
        #print (output)
        output = cur.fetchall()
        if output == []:
            print("|| Информация не найдена")
        else:
            print("|| ", name_2, name_3, "преподает в:")
            for row in output:
                print ("\t", '%-4s' % (row[0]))
            os.system('pause')
    except sq.Error as e:
        if db: db.rollback()
        print("|| Ошибка выполнения запроса ", e)
        os.system('pause')
    finally:
        if db: db.close()

def key7():
    try:
        print("|| Формирование справки о количестве учеников в классе\n")
        db = sq.connect("school.db")
        cur = db.cursor()
        cur.execute("""Begin;""")
        doc = Document()

```

```

style = doc.styles['Normal']
style.font.name = 'Times New Roman'
style.font.size = Pt(14)
head = doc.add_heading("Справка о количестве учеников", level=0)
head.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER
print("|| Введите класс")
grade = input(" ")
doc_text = doc.add_paragraph('B ')
doc_text.add_run(grade).bold = True
doc_text.add_run(' учатся ')
request = """
    SELECT count(id_student) as "Количество учеников"
    FROM id_students
    WHERE grade = ?
"""
cur.execute(request, (grade,))
#print (output)
output = cur.fetchall()
if output == []:
    print("|| Такого класса нет")
else:
    for row in output:
        print ("\t", '%-5s' % (row[0]))
        print("|| Количество учеников в", grade)
        doc_text.add_run(str(row[0]))
        doc_text.add_run(' учеников.')
        doc.save("StudentsCount.docx")
    os.system('pause')
except sq.Error as e:
    if db: db.rollback()
    print("|| Ошибка выполнения запроса ", e)
    os.system('pause')
finally:
    if db: db.close()
def key8():
    try:
        print("|| Формирование отчета о работе школы\n")
        db = sq.connect("school.db")
        cur = db.cursor()
        cur.execute("""Begin;""")
        doc = Document()
        style = doc.styles['Normal']
        style.font.name = 'Times New Roman'
        style.font.size = Pt(14)
        head = doc.add_heading("Отчет о работе школы", level=0)
        head.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER
        #1 - количество учителей по предметам
        doc_text = doc.add_paragraph('Количество учителей по предметам:')
        doc.add_paragraph()
        request = """
            SELECT subject as "Предмет", count(DISTINCT id_teacher) as "Количество учителей"
            FROM teachers group by subject;
        """
        cur.execute(request)
        #print (output)
        output = cur.fetchall()
        table = doc.add_table(rows=1, cols=2)
        cells = table.rows[0].cells
        cells[0].text = 'Предмет'
        cells[1].text = 'Количество учителей'

```

```

for subj, n_teach in output:
    #print ("\t", '%-15s %-5s' % (subj,n_teach))
    cells = table.add_row().cells
    cells[0].text = subj
    cells[1].text = str(n_teach)
doc.add_paragraph()
#2 - количество кабинетов
doc_text = doc.add_paragraph('Количество кабинетов в школе: ')
request = """
    SELECT count(DISTINCT room) FROM teachers
    WHERE not (room = "");
"""
cur.execute(request)
#print (output)
output = cur.fetchall()
for row in output:
    #print ("\t", '%-5s' % (row[0]))
    doc_text.add_run(str(row[0]))
doc_text.add_run('.')
doc.add_page_break()
#3 - Количество учителей в каждом классе
doc_text = doc.add_paragraph('Количество учителей на класс: ')
doc.add_paragraph()
request = """
    select grade as "Класс", count(DISTINCT id_teacher) as "Кол-во учителей"
    from teachers group by grade;
"""
cur.execute(request)
#print (output)
output = cur.fetchall()
table = doc.add_table(rows=1, cols=2)
cells = table.rows[0].cells
cells[0].text = 'Класс'
cells[1].text = 'Количество учителей'
for grade, n_teach in output:
    #print ("\t", '%-5s %-5s' % (grade, n_teach))
    cells = table.add_row().cells
    cells[0].text = grade
    cells[1].text = str(n_teach)
#4 - количество двоечников
doc_text = doc.add_paragraph('Количество двоечников в школе: ')
request = """
    SELECT count(DISTINCT id_student) as "Кол-во учеников" from students
"""
cur.execute(request)
#print (output)
output = cur.fetchall()
for row in output:
    #print ("\t", '%-5s' % (row[0]))
    doc_text.add_run(str(row[0]))
doc_text.add_run('.')
#5 - количество хорошистов
doc_text = doc.add_paragraph('Количество хорошистов в школе: ')
request = """
    SELECT count(DISTINCT id_student) as "Кол-во учеников" from students
"""
cur.execute(request)
#print (output)
output = cur.fetchall()
for row in output:

```



```

        #print ("\t", '%-5s' % (row[0]))
        doc_text.add_run(str(row[0]))
    doc_text.add_run('.')
    #6 - количество отличников
    doc_text = doc.add_paragraph('Количество отличников в школе: ')
    request = """
        SELECT count(DISTINCT id_student) as "Кол-во учеников" from students
    """

    cur.execute(request)
    #print (output)
    output = cur.fetchall()
    for row in output:
        #print ("\t", '%-5s' % (row[0]))
        doc_text.add_run(str(row[0]))
    doc_text.add_run('.')
    doc.save("School.docx")
    os.system('pause')
except sq.Error as e:
    if db: db.rollback()
    print("|| Ошибка выполнения запроса ", e)
    os.system('pause')
finally:
    if db: db.close()

def key9():
    print("Создать резервную копию БД или восстановить БД из файла?\n")
    print("1 - Создать")
    print("2 - Восстановить")
    print("0 - Вернуться назад\n")
    key = input()
    while key != '0':
        if key == '1':
            os.system('cls')
            sq_dump_w()
            print("Резервная копия создана\n")
            os.system('pause')
            break
        if key == '2':
            os.system('cls')
            sq_dump_r()
            print("Резервная копия восстановлена\n")
            os.system('pause')
            break

def main():
    sq_init()
    choose_user()
    user = input()
    choose_main(user)
    key = input()
    while key != '0':
        if key == '1':
            os.system('cls')
            key1()
            choose_main(user)
            key = input()
        if key == '2' and user == '2':
            os.system('cls')
            key2()
            choose_main(user)

```

```

        key = input()
    if key == '3':
        os.system('cls')
        key3()
        choose_main(user)
        key = input()
    if key == '4' and user == '2':
        os.system('cls')
        key4()
        choose_main(user)
        key = input()
    if key == '5':
        os.system('cls')
        key5()
        choose_main(user)
        key = input()
    if key == '6' and user == '2':
        os.system('cls')
        key6()
        choose_main(user)
        key = input()
    if key == '7' and user == '2':
        os.system('cls')
        key7()
        choose_main(user)
        key = input()
    if key == '8' and user == '2':
        os.system('cls')
        key8()
        choose_main(user)
        key = input()
    if key == '9' and user == '2':
        os.system('cls')
        key9()
        choose_main(user)
        key = input()
    return 0

main()

```