

МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Комплексной безопасности ТЭК

Кафедра Управления безопасностью сложных систем

ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

по дисциплине Технологии и методы программирования

на тему Разработка программного приложения «Школа»

ДАНО
студентам

Батовой Полине Александровне

Гаврилову Евгению Сергеевичу

Летуновскому Игорю Олеговичу

(фамилия, имя, отчество)

группы

КА-20-03

(номер группы)

Содержание работы:

1. Описание требований к разрабатываемому приложению
2. Разработка программного кода
3. Тестирование приложения

Исходные данные для выполнения работы:

1. Информация о предметной области из открытых источников информации

Рекомендуемая литература:

1. Положение о курсовом проектировании, утв. приказом от 11.11.2021 № 398
2. Тузовский, А.Ф. Объектно-ориентированное программирование: учебное пособие для вузов / А.Ф. Тузовский. — Москва: Издательство Юрайт, 2022. — 206 с. — (Высшее образование). — ISBN 978-5-534-00849-4. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/490369>.

Графическая часть:

1. Скриншоты форм приложения
2. Листинг программного кода

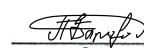


Руководитель: канд. физ.-мат. наук
(уч. степень)

доцент
(должность)


(подпись)

Малашкин А.В.
(фамилия, имя, отчество)

Задание принял к исполнению: студент




(подпись)

Батова П.А.

Гаврилов Е.С.

Летуновский И.О.

(фамилия, имя, отчество)

МИНОБРНАУКИ РОССИИ

РГУ НЕФТИ И ГАЗА (НИУ) ИМЕНИ И.М. ГУБКИНА

Факультет Комплексной безопасности ТЭК

Кафедра Управления безопасностью сложных систем

Оценка комиссии: _____ Рейтинг: _____

Подписи членов комиссии:

(подпись)

(фамилия, имя, отчество)

(подпись)

(фамилия, имя, отчество)

(дата)

КУРСОВАЯ РАБОТА

по дисциплине _____ Технологии и методы программирования

на тему _____ Разработка программного приложения «Школа»

«К ЗАЩИТЕ»

доцент, канд. физ.-мат. наук

Малашкин А.В.

(должность, ученая степень; фамилия, и.о.)



(подпись)

(дата)


ВЫПОЛНИЛИ:

Студенты группы _____

КА-20-03

(номер группы)

Батова П.А.



Гаврилов Е.С.



Летуновский И.О.



(фамилия, имя, отчество)

(подпись)

(дата)

Москва, 2022

Содержание

Введение	3
1. Спецификация	5
1.1. Анализ предметной области, построение модели.....	5
1.2. Описание классов, методов, алгоритмов взаимодействия.....	6
1.3. Разработка структурной схемы интерфейса.....	7
2. Разработка программного кода приложения.....	9
2.1. Разработка классов.....	9
2.2. Разработка интерфейса приложения.....	12
3. Тестирование приложения.....	37
3.1. Тест-план.....	37
3.2. Карта приложения.....	39
3.3. Создание чек-листов.....	40
3.4. Создание тест-кейсов.....	41
3.4. Определение дефектов.....	50
3.5. Результаты тестирования.....	52
Заключение.....	53
Список используемой литературы.....	55

Введение

Школьный журнал – необходимая для каждой школы форма хранения информации о классах, учениках, их оценках и посещаемости. Для каждого класса ведется собственный журнал. До недавнего времени школьный журнал существовал только в бумажном формате. Однако с развитием информационных технологий стало ясно, что гораздо удобнее пользоваться журналом в электронном виде.

Электронные журналы обладают рядом преимуществ перед печатными аналогами. Во-первых, надежность. Бумажный вариант можно испортить или потерять (особенно с учетом того, что обычно печатные журналы носят с урока на урок сами ученики), электронный же не подвержен этим рискам. Хранение данных на компьютерах позволяет легко делать периодически резервные копии и хранить их в облачных хранилищах, что несравнимо проще чем печатать дополнительные запасные журналы и время от времени заполнять еще и их. Во-вторых, в электронной версии легко исправлять оценки и другую информацию об учениках. В-третьих, электронный журнал может автоматически высчитывать оценку ученика и вести учет посещаемости. В-четвертых, в современных журналах существует функция автоматического перевода учащихся в следующий класс. В-пятых, руководству гораздо проще проверять журналы на компьютере, переходя от класса к классу лишь нажатием кнопки, чем вручную просматривать каждый отдельный журнал.

Таким образом, электронная версия школьного журнала упрощает работу учителям и руководству и является полезной технологией для любой школы.

Целью нашей курсовой работы является получение практических навыков в создании приложений с использованием высокоуровневых методов программирования в процессе разработки программы «Школа», поддерживающей режимы учета классов и учеников в них, добавление и удаление учеников, учет посещаемости занятий и оценок учащихся, то есть

фактически выполняющее функции обычно школьного журнала, на языке C++ в среде разработки Qt Creator.

Для этого перед нами стоят следующие задачи: анализ предметной области, построение модели программного обеспечения, описание классов, методов и алгоритмов взаимодействия, разработка структурной схемы интерфейса, разработка классов, разработка интерфейса приложения, тестирование приложения, в частности разработка кейс-тестов и анализ результатов проведения тестирования.

1. Спецификация

1.1 Анализ предметной области, построение модели

Целью программного обеспечения является представление школьного журнала и выполнение его функций.

Данная программа пригодится учителям как альтернатива бумажному варианту школьного журнала. Электронный журнал будет удобно хранить все нужные данные (оценки, неявки, имена учеников), автоматически вычислять средний балл и процент посещений.

Для использования ПО пользователю необходим хотя бы минимальный уровень знания ПК.

Функции программы:

- Просмотр оценок учеников по выбранному предмету
- Редактирование списка учеников в выбранном классе (добавить, удалить [при удалении должно выводиться предупреждение], изменить)
- Редактирование оценок ученика в выбранном классе и по выбранному предмету (добавить, удалить, пропуск занятия)
- Поиск ученика по имени и фамилии (по выбранному классу, либо по всей школе). Поиск осуществляется либо только по имени, либо только по фамилии, при наличии ошибки в вводе, ищется ближайшее совпадение
- Переход к следующему году (оставляет на второй год учеников с посещаемостью ниже 50% или средней оценкой ниже 2,5 по любому из предметов; остальные ученики переходят в следующий класс)
- Вычисление средних оценок по предметам
- Вычисление процента посещаемости

Модель ПО:

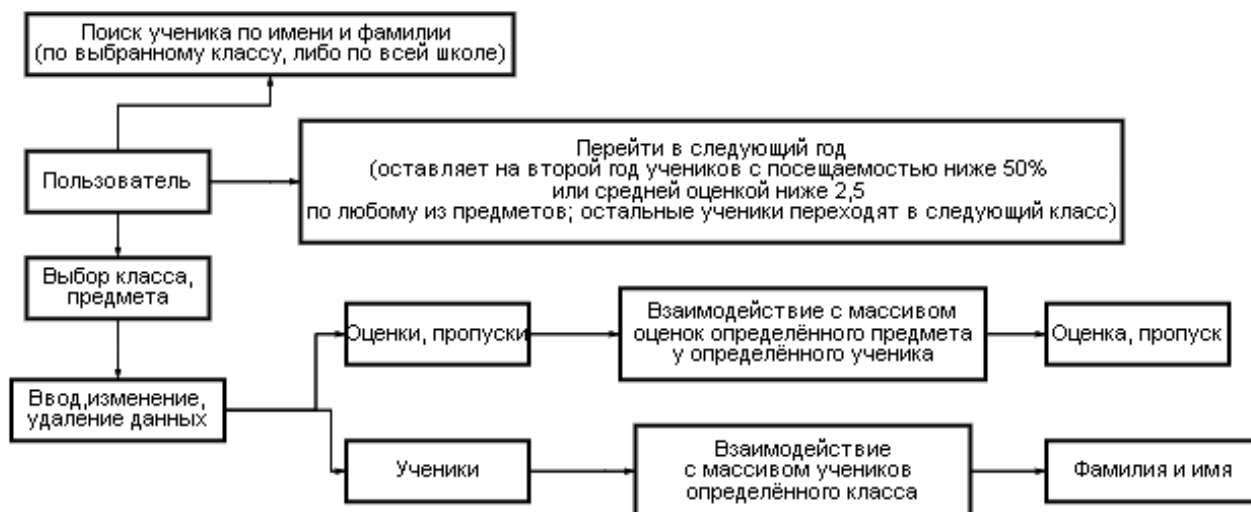


Рис 1. Модель ПО

1.2. Описание классов, методов, алгоритмов взаимодействия

В состав входит 2 класса: класс, ученик.

В класс “класс” входит массив учеников, в котором максимальное количество учащихся – 30, а также атрибут номер класса.

В класс “ученик” входит двумерный массив (первая ячейка отвечает за предмет, вторая – за оценку на конкретном занятии), одномерный массив со средними оценками по всем предметам, атрибут имя и атрибут фамилия.

Названия классов (например “1”, “3”), а также названия предметов (физкультура, математика и т.д.) будут заранее находиться в данных школы. У разных классов – разные предметы.

Ограничения программы:

Программа должна хранить информацию об 11 классах (с 1 по 11 класс без учета буквы), максимальное количество учеников в классе – 30 человек.

Перечень предметов для классов должен отличаться. Таким образом, в 1 классе есть предметы: русский язык, чтение, физкультура, музыка, математика, ИЗО и окружающий мир.

С 2 по 4 класс ученики изучают русский язык, чтение, физкультуру, музыка, математику, ИЗО, окружающий мир и английский язык.

С 5 по 6 ученики изучают русский язык, литературу, физкультуру, математику, ИЗО, географию, историю, обществознание, биологию и английский язык.

С 7 по 11 ученики изучают русский язык, литературу, физкультуру, алгебру, геометрию, ИЗО, географию, историю, обществознание, биологию, английский язык, химию, информатику и физику.

1.3. Разработка структурной схемы интерфейса.

При запуске программы будут создаваться 2 формы:

Первая - основное приложение, появляется на экране при запуске

Вторая вызывается при нажатии кнопки “новый ученик” или “редактировать имя” в первой форме. В ней вводится имя и фамилия ученика.

2 → 5 →

3 → 6 → 7 ↑

4 → 8 →

1: Таблица данных
2: Класс
3: Новый ученик
4: Удаление ученика
5: Предмет
6: Редактировать имя
7: Поиск
8: Начало нового учебного года

Рис 2. Модель интерфейса (первая форма)

Введите имя и фамилию
нового ученика

Имя

Фамилия

1, 2:
Поля для
ввода

ОК ОТМЕНА

Рис 3. Модель интерфейса (вторая форма)

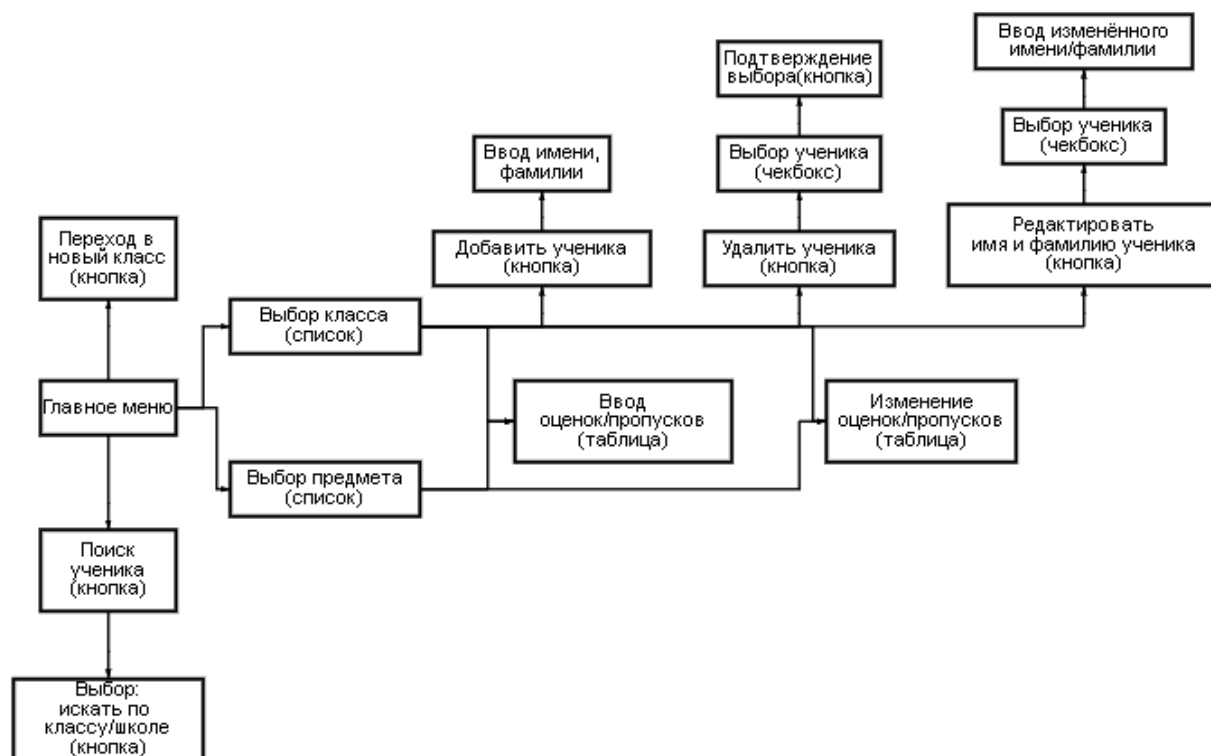


Рис 4. Структурная схема интерфейса

2. Разработка программного кода приложения

2.1. Разработка классов

Класс – это, условно говоря, пользовательский тип данных, чаще всего многомерный. При создании переменной такого типа создается экземпляр класса – то есть объект. В описании класса хранятся свойства и поведение объекта. Свойства – это его атрибуты, переменные, записанные в теле класса, и присущие каждому объекту этого класса. Под поведением объекта обычно имеют ввиду методы, которые можно выполнять с его данными.

Базовым классом в нашем приложении является ученик – «Student», поскольку цель любого журнала – хранить информацию об учащихся.

```
class Student{
private:
    QString name = ""; //имя ученика, по умолчанию пусто
    QString surname = ""; //фамилия ученика, по умолчанию пусто
public:
    Student(QString Name) {...}
    Student(QString Name,QString Surn) {...}
    Student() {...} //конструктор по умолчанию
    QString getName {...} //получить имя
    void setName(QString Name) {...} //изменить имя
    QString getSurn() {...} //получить фамилию
    void setSurn(QString sName) {...} //изменить фамилию
    double middleGrade[subj_num]={6,6,6,6,6,6,6,6,6,6,6,6,6,6,6};
//средние оценки
    double absence=0; //посещаемость
    bool isViable = true; //можно ли переводить на следующий год
    int gradeSum = 0; //сумма оценок
    int gradeNum = 0; //количество занятий
    int gradeNumTotal=0;
    int missNum = 0; //количество прогулов
    QString grades[subj_num][lesn_num]; //массив всех оценок
    //обновление информации об ученике
    void updateQuality() {...};
```

Листинг 1. Класс «Ученик».

Средние оценки заранее инициализированы невозможным значением (в пятибалльной системе оценивания) для того, чтобы они не отображались в интерфейсе программе, о чем будет в следующей главе.

Логическая переменная `isViable` отвечает за возможность перевода конкретного ученика на следующий год. Если эта переменная равна нулю, то при начале следующего учебного года, учащийся останется в том же классе.

У базового класса есть 3 конструктора:

```
Student(QString Name) //конструктор, устанавливающий имя ученика
{ this->name = Name; }
Student(QString Name,QString Surn) //конструктор, устанавливающий
фамилию ученика
{ this->name=Name;
  this->surname=Surn;}
Student() //конструктор по умолчанию
{ }
```

Листинг 2. Конструкторы класса «Ученик».

Методы базового класса:

У базового класса есть типичные методы сеттеры и геттеры. Методы `setName(QString Name)` и `setSurn(QString sName)` устанавливают имя и фамилию ученика соответственно. Методы `QString getName` и `QString getSurn()` возвращают значение имени и фамилии соответственно.

Метод `updateQuality()` обновляет информацию об ученике. В частности, считает среднее арифметическое по всем предметам, ведет учет посещаемости, проверяет, можно ли переводить учащегося на следующий год (условиями для этого являются посещаемость более 50% и отсутствие двоек по всем предметам).

```
void updateQuality() {
    this->isViable=true;
    this->missNum=0;
    this->gradeNumTotal=0;
    for(int i=0;i<subj_num;i++) {
        this->gradeSum=0;
        this->gradeNum=0;
        for(int j=0;j<lesn_num;j++)
        {
            if (this->grades[i][j][0].isDigit())
//проверка, что оценка действ. число
            {
                this->gradeSum+=this-
>grades[i][j].toInt(); //подсчет суммы оценок
                this->gradeNum+=1; //подсчет кол-ва
оценок
            } else if (this->grades[i][j]=="н"&&this-
>grades[i][j]!="") //если находит пропуск
```

```

        this->missNum+=1; //то кол-во пропусков +1
    }
    this->gradeNumTotal+=this->gradeNum;
    if (this->gradeNum!=0)
        this->middleGrade[i]=(double) this-
>gradeSum/this->gradeNum; //подсчет средней оценки
        if (this->middleGrade[i]<2.5)
            this->isViable=false;}
        if (this->missNum+this-> gradeNumTotal!=0) //проверка,
есть ли вообще информация о пропусках/оценках
            this->absence=(double) (1-(double) ((double) this-
>missNum/ (double) (this->missNum+this->gradeNumTotal))); //если
есть считает посещаемость
            if (this->absence<0.5)
                this->isViable=false;
    }
};

```

Листинг 3. Реализация метода проверки, можно ли переводить учащегося на следующий год.

В результате работы метода переменная isViable принимает значение 0 или 1. В циклах for, которые проходят в первом случае по предметам, а во втором по занятиям, для задачи условия выхода используются константные значения:

```

const int subj_num = 14; //количество предметов
const int lesn_num = 14; //кол-во занятий

```

Листинг 4. Объявление констант количество предметов и количество занятий.

Вторым классом в приложении является «Класс» - «Group». Его реализация довольно проста:

```

class Group{
    public:
        int name;
        Student students[stud_num];
};

```

Листинг 5. Класс «класс».

Для создания массива на 30 учеников используется константное значение:

```

const int stud_num = 30; //макс количество учеников в классе

```

Листинг 6. Объявление константы – максимальное количество учеников в классе.

```

const int grup_num = 11; //количество классов

```

Листинг 7. Объявление константы – количество классов.

Для работы в программе был объявлен массив классов:

```
Group groups [grup_num] ;
```

Листинг 8. Создание массива классов.

2.2. Разработка интерфейса

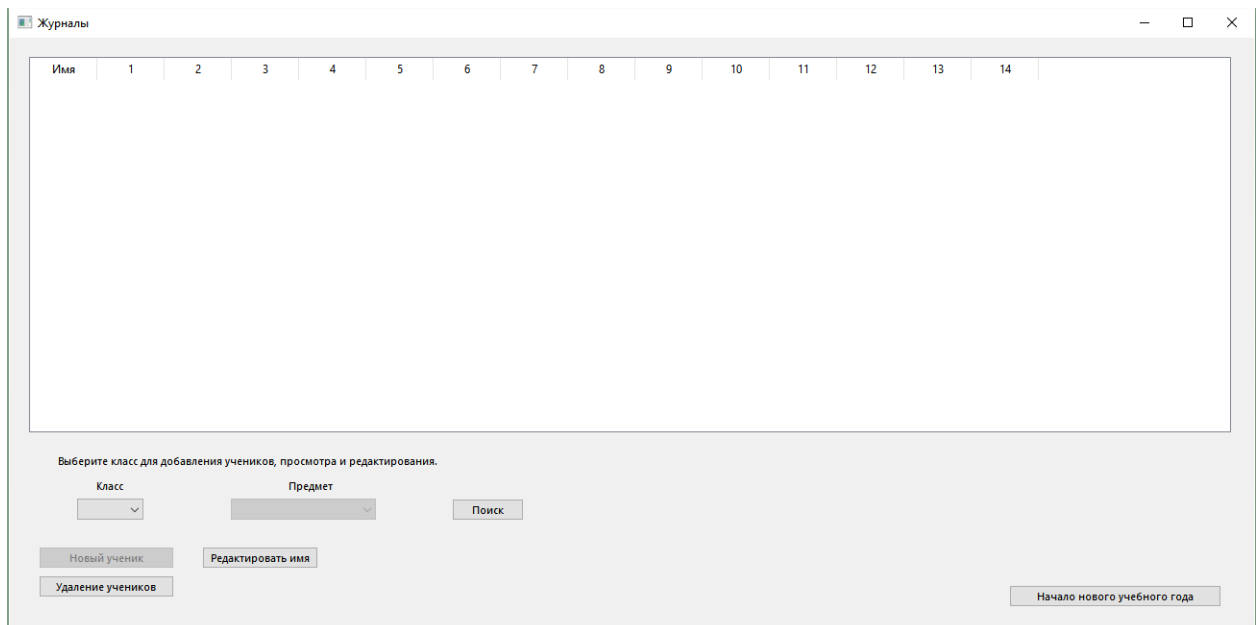


Рис. 5. Интерфейс основной формы.

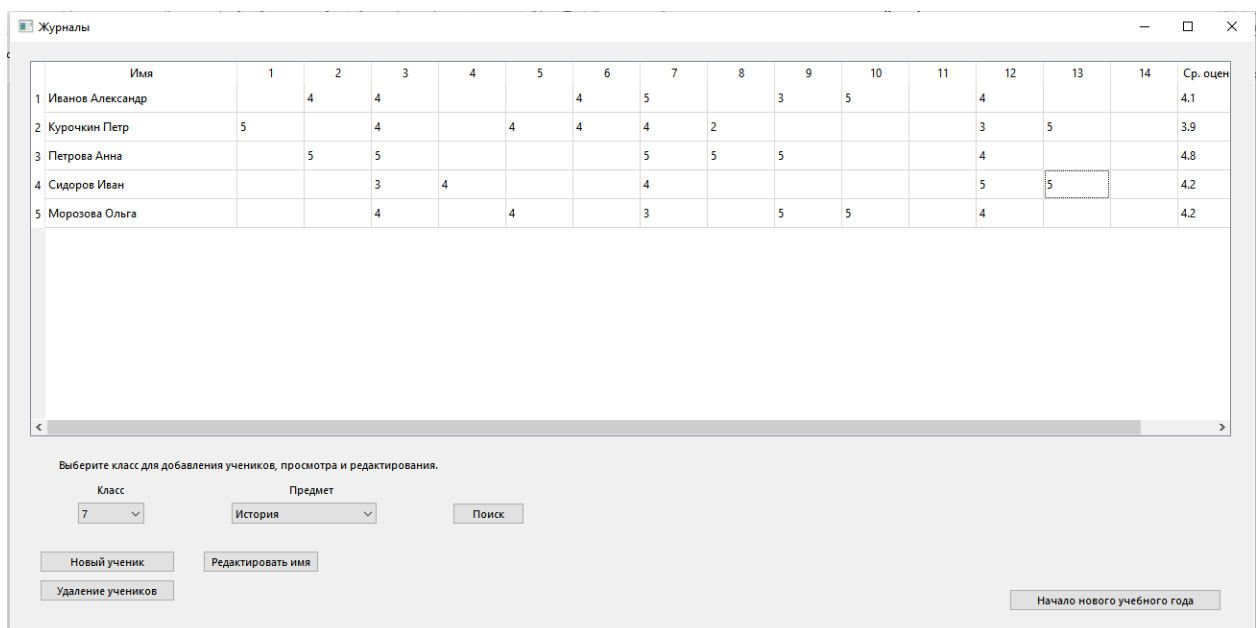


Рис. 6. Пример работы программы.

В основном файле `main.cpp` выполняется вызов основного окна приложения.

```
#include "widget.h" //подключение реализации окна
#include <QApplication>
int main(int argc, char *argv[])
{
```

```

        QApplication a(argc, argv); //объявление приложения
        класса QApplication
        Widget w; //объявление объекта класса Widget,
определенного в widget.cpp
        w.show(); //показ виджета
        return a.exec(); //запуск
    }

```

Листинг 8. Main.cpp.

Заголовочный файл `widget.h` включает в себя реализацию класса окна приложения. Оно наследует методы встроенного класса `QWidget`, а также включает в себя объявления всех функций окна (сигналы и слоты), включая конструктор и деструктор, объявления всех пользовательских функций для `Widget`. Здесь же создается экземпляр класса `Widget` во внешней области видимости `UI`.

В файле `widget.cpp` располагается реализация класса `Widget`. Поскольку внешний вид программы был создан автоматически с помощью конструктора GUI приложений `Qt Designer`, то был создан сгенерированный файл `widget.ui`, содержащий в себе дизайн приложения и инициализацию экземпляров классов элементов приложения. Такими классами являются `QTableWidget` (основная таблица), `QPushButton` (кнопки), `QComboBox` (выпадающие списки), `QLabel` (строки не редактируемого текста).

В приведенном ниже фрагменте кода определяются конструкторы и деструкторы виджета. Конструктор собирает все, что создано в `widget.ui`, а далее сразу скрывает первую колонку таблицы `QTableWidget tableWidget` с помощью метода `hideColumn(index)`.

```

Widget::Widget(QWidget *parent)
    : QWidget(parent)
    , ui(new Ui::Widget)
{
    ui->setupUi(this);
    ui->tableWidget->hideColumn(0);
}
Widget::~Widget()
{
    delete ui;
}

```

Листинг 9. Конструктор и деструктор основного окна.

По умолчанию список выбора предмета и кнопка добавления ученика недоступны, остальные кнопки не будут иметь эффекта, поэтому сначала выбирается класс в выпадающем списке `comboBox`. При выборе элемента из списка (с 1 по 11 класс) происходит вызов функции `on_comboBox_activated`, которая получает на вход индекс `index` выбранного элемента.

```
void Widget::on_comboBox_activated(int index)
{
    currentGroup = index;
    ui->tableWidget->hideColumn(0);
    if (currentGroup != -1) {
        ui->pushButton->setEnabled(true);
        ui->pushButton->setEnabled(true);
        ui->pushButton_2->setEnabled(true);
        ui->pushButton_3->setEnabled(true);
        ui->pushButton_5->setEnabled(true);
        ui->comboBox_2->setEnabled(true);
        ui->comboBox_2->clear();
        switch(currentGroup) {
            case 0 : {ui->comboBox_2->addItem(one);break;}
            case 1: case 2: case 3: {ui->comboBox_2->
>addItem(twotofour);break;}
            case 4: case 5: {ui->comboBox_2->addItem(fivetosix);break;}
            case 6: case 7: case 8: case 9: case 10: {ui->comboBox_2->
>addItem(seventoeleven);break;}
        }
        ui->comboBox_2->setCurrentIndex(-1);
        currentSubject=-1;
    }
    else{
        ui->pushButton->setEnabled(false);
        ui->comboBox_2->setEnabled(false);
    }
    setupTables();
}
```

Листинг 10. Функция выпадающего списка выбора класса.

Сначала присваивается значение глобальной переменной `currentGroup`, отвечающей за выбранный класс. Если произошел выбор, а не сброс (присваивая -1), функция включает все остальные кнопки методом `setEnabled(bool)` и с помощью конструкции `switch` от значения текущего класса устанавливает в `comboBox_2` новые значения элементов. Для этого используются заранее заготовленные списки `QStringList`, а также очистка

элементов комбобокса методом `clear()` и их добавление методом `addItem(QStringList)`.

```
QStringList one = {"Русский язык", "Чтение", "Физ-  
ра", "Музыка", "Математика", "ИЗО", "Окр. мир"};  
QStringList twotofour = {"Русский язык", "Чтение", "Физ-  
ра", "Музыка", "Математика", "ИЗО", "Окр. мир", "Англ. язык"};  
QStringList fivetosix = {"Русский язык", "Литература", "Физ-  
ра", "Математика", "ИЗО", "География", "История", "Обществозн.", "Биол  
огия", "Англ. язык"};  
QStringList seventoeleven = {"Русский  
язык", "Литература", "Физ-  
ра", "Алгебра", "Геометрия", "ИЗО", "География", "История", "Обществозн.  
н.", "Биология", "Англ. язык", "Химия", "Информатика", "Физика"};
```

Листинг 11. Создание списков предметов для разных классов.

Часть строчек используется преимущественно для придания вида формы или сброса значений во избежание ошибок. В конце вызывается функция `setupTables()`, которая призвана настроить таблицу под выбранный класс.

```
void Widget::setupTables ()  
{  
    if (currentGroup!=-1)  
    {  
        formateTable();  
        int i=0;  
        ui->tableWidget->clearContents();  
        ui->tableWidget->setRowCount(0);  
        while (groups[currentGroup].students[i].getName() != "") {  
            ui->tableWidget->setRowCount(ui->tableWidget->  
>rowCount()+1);  
            QTableWidgetItem *item = new QTableWidgetItem;  
            item->  
>setText(groups[currentGroup].students[i].getSurn()+"  
"+groups[currentGroup].students[i].getName());  
            item->setFlags(item->flags() & ~Qt::ItemIsEditable);  
            ui->tableWidget->setItem(ui->tableWidget->  
>rowCount()-1,1,item);  
            if(currentSubject==-1){  
                for(int j =2;j<ui->tableWidget->columnCount()-  
1;j++){  
                    if  
(groups[currentGroup].students[i].middleGrade[j-2]!=6)  
                        ui->tableWidget->setItem(i,j,new  
QTableWidgetItem(QString::number(groups[currentGroup].students[i]  
].middleGrade[j-2],'f',1));  
                }  
                ui->tableWidget->setItem(i,ui->tableWidget->  
>columnCount()-1,new QTableWidgetItem(QString("%1
```



```

%").arg(QString::number(groups[currentGroup].students[i].absence
*100,'f',1)))));
    }
    i++;
}
restrictEdit();
}
}

```

Листинг 12. Вставка имен и вида окна при выбранном классе.

Если выбор класса совершен, происходит заполнение таблицы элементами. Сначала она полностью очищает элементы таблицы методом `clearContents()` и убирает все строки методом `setRowCount(int)`, а затем, соответственно выбранному классу, выводит все имена учеников, добавляя строчки. Делается это в цикле `while` пока не найдется пустой ученик с именем "". Сначала добавляется строчка тем же методом `setRowCount()`, используя текущее количество строк, возвращаемое методом `rowCount()`. Для добавления элемента в таблицу необходимо создать объект класса `QTableWidgetItem`, содержащий все свойства того, что должно быть в ячейке. В данном случае объект `item` создается с использованием `new`. Затем методом `setText(QString)` ему меняется текстовое наполнение, в данном случае фамилия и имя. Здесь же с помощью флагов объекта и метода `setFlags()` меняется флаг `Qt::ItemIsEditable`, отвечающий за редактируемость ячейки. В скобках метода используется битовая операция, согласно которой текущие значения флагов объекта (получаемые методом `flags()`) умножаются на отрицательное значение флага `Qt::ItemIsEditable`, что гарантирует отключение флага. Методом `setItem(row,column,item)` устанавливается объект в ячейку. Если учеников нет, то функция ничего не сделает. Далее после проверки, не выбран ли предмет, он заполняет остальные ячейки строки средними оценками соответственно формату таблицы и предметов, соответственно классу ученика. Делается это тем же методом `setItem`, однако без предварительного создания объекта с помощью конструктора `new QTableWidgetItem(QString)`. Поскольку среднее арифметическое имеет тип `double`, а для текста необходим `QString`, то вызывается функция

number(double,format,precision) из области видимости QString. В последний столбец выводится посещаемость ученика по всем предметам с помощью такого же способа, однако добавляется использование интерполяции строк QString().args(QString), хотя его использование здесь не обязательно. В конце все ячейки становятся не редактируемыми с помощью метода restrictEdit().

Помимо этого в этой же функции вызывается другая пользовательская функция formatTable().

```
void Widget::formatTable () {
    if (currentGroup!=-1&&currentSubject==-1) {
        int i=0;
        ui->tableWidget->setColumnCount(2);
        QStringList list;
        switch (currentGroup) {
        case 0 : {list=one;break;}
        case 1: case 2: case 3: {list=twotofour;break;}
        case 4: case 5: {list=fivetosix;break;}
        case 6: case 7: case 8: case 9: case
10:{list=seventoeleven;break;}
        }
        while (list.size()>i)
        {
            if (ui->tableWidget->columnCount()<=i+2)
                ui->tableWidget->setColumnCount(ui->tableWidget-
>columnCount()+1);
            ui->tableWidget->setHorizontalHeaderItem(2+i, new
QTableWidgetItem(QString("Сп. оценка\n%1").arg(list[i])));
            ui->tableWidget->setColumnWidth(i+2,85);
            i++;
        }
        if (ui->tableWidget->columnCount()<=i+2)
            ui->tableWidget->setColumnCount(ui->tableWidget-
>columnCount()+1);
        ui->tableWidget->setHorizontalHeaderItem(2+i, new
QTableWidgetItem("Посещаемость"));
        ui->tableWidget->setColumnWidth(i+2,100);
        ui->tableWidget->setColumnWidth(1,200);
        }
        if (currentSubject!=-1) {
            ui->tableWidget->setColumnCount(2);
            for (int i =1;i<16;i++){
                if(i!=15){
                    if (ui->tableWidget->columnCount()<=i+1)
                        ui->tableWidget->setColumnCount(ui-
>tableWidget->columnCount()+1);
                    ui->tableWidget-
>setHorizontalHeaderItem(1+i, new
QTableWidgetItem(QString::number(i)));
                }
            }
        }
    }
}
```

```

        }
        else {
            if (ui->tableWidget->columnCount() <= i+1)
                ui->tableWidget->setColumnCount(ui-
>tableWidget->columnCount()+1);
            ui->tableWidget-
>setHorizontalHeaderItem(1+i, new QTableWidgetItem("Ср.
оценка"));
        }
    }
}

if (currentGroup==-1 && currentSubject==-1)
    ui->tableWidget->setColumnCount(2);
}

```

Листинг 13. Форматирование размера таблицы и ее заголовков.

Эта функция призвана менять количество столбцов и их заголовки в зависимости от положения комбобоксов. Если выбран класс, но не выбран предмет, то в зависимости от класса с помощью switch определяется нужный список элементов второго выпадающего списка. Пока список не кончится (определяя размер списка с помощью size()) создаются новые колонки и устанавливаются их заголовки с помощью setHorizontalHeaderItem(column, item), который практически идентичен setItem(), однако не нужно указывать ряд. При этом меняется ширина столбцов с помощью setColumnWidth(column, width). После этого меняется последняя колонка с посещаемостью аналогично.

Если же предмет выбран, то применяется совершенно другое форматирование. Здесь функция просто нумерует столбцы, чтобы было количество столбцов для заполнения их оценками равно const lesn_num. Последний столбец отводится под среднюю оценку.

Если ни класс, ни предмет не выбран, то оставляются лишь первые две колонки (из которых по умолчанию первая скрыта).

Далее после выбора класса необходимо добавлять учеников соответствующей кнопкой. По ее нажатию вызывается функция on_pushButton_clicked.

```

void Widget::on_pushButton_clicked()
{
    ndlg = new NameDialog();
}

```

```

connect(ndlg,SIGNAL(sendData(QString,QString)),this,SLOT(receive
Data(QString,QString)));
    ndlg->exec();
    if (receivedname!="")
    {
        if(ui->tableWidget->rowCount()<30){
            QString text=receivedsum+" "+receivedname;
            ui->tableWidget->setRowCount(ui->tableWidget->
            >rowCount()+1);
            QTableWidgetItem *item = new QTableWidgetItem;
            item->setText(text);
            item->setFlags(item->flags() & ~Qt::ItemIsEditable);
            ui->tableWidget->setItem(ui->tableWidget->rowCount()-
            1,1,item);
        }
        else createmsgSignal("Вы ввели максимальное
количество учеников!");
    }
}

```

Листинг 14. Добавление ученика в таблицу.

Этой функцией создается новый экземпляр объекта класса NameDialog, который является второй пользовательской формой приложения, содержащей в себе поля для ввода имени и фамилии. Далее используется конструктор connect(sender,SIGNAL,receiver,SLOT) для создания связи между двумя функциями двух разных объектов. В данном случае функция считывания имени и фамилии в диалоге связывается с приемом этих данных основной формой. Сигнал – функция, вызывающая слот. Слот receiveData() присваивает глобальным переменным receivedname и receivedsum полученные имя и фамилию. Если что-то было получено в результате выполнения формы диалога с помощью exec(), то проверив заполненность таблицы, добавляется новая строка с именем и фамилией учеников в 1 колонке, иначе выводит QMessageBox с указанным сообщением.

Добавление ученика производится не напрямую через кнопку добавления, а косвенно внутри тела функции on_tableWidget_cellChanged, которая вызывается всякий раз, когда значение ячейки меняется. Логика устроена таким образом, потому что таблица хорошо отображает использующуюся структуру массива и при обращении к его элементу можно использовать индексы строк и столбцов.

```

void Widget::on_tableWidget_cellChanged(int row, int column)
{
    if (column==1 && currentGroup!=-1) {
        if
(groups[currentGroup].students[row].getName()=="") {

groups[currentGroup].students[row]=Student (receivedname, received
urn);

        receivedname="";
        receivedsurn="";
        }
    }
    else if (column>1 && column!=16 && currentSubject !=-
1&&ui->tableWidget->item(row,column)-
>text().size()<2&&checkGrade(ui->tableWidget->item(row,column)-
>text())) {

groups[currentGroup].students[row].grades[currentSubject][column
-2]=ui->tableWidget->item(row,column)->text();
        groups[currentGroup].students[row].updateQuality();
        if
(groups[currentGroup].students[row].middleGrade[currentSubject]!
=6) {
            ui->tableWidget->setItem(row,16,new
QTableWidgetItem(QString::number(groups[currentGroup].students[r
ow].middleGrade[currentSubject], 'f',1));
            ui->tableWidget->item(row,16)->setFlags(ui-
>tableWidget->item(row,16)->flags() &~ Qt::ItemIsEditable);
        }
    }
    else if (column!=16&&column>1&&currentSubject!=-1) ui-
>tableWidget->item(row,column)->setText("");
}

```

Листинг 15. Изменение ячейки, фрагмент о добавлении оценки.

Если колонка, где было произведено изменение первая, была выбрана группа и строка, куда вбивается ученик, уже не содержит ученика по адресу, то создается новый ученик с помощью конструктора Student(QString, QString), а глобальные переменные-приемники обнуляются. Если изменение происходит в колонках после первой (кроме последней), был выбран предмет, а так же у введенного текста прошла проверка на правильность (количество символов должно быть меньше 2 (метод size()), это должна быть цифра от 2 до 5 или буква «н» (функция checkGrade(QString))), то происходит добавление оценки. Здесь же сразу же по изменении оценки вызывается метод updateQuality(), который пересчитывает средние оценки и

посещаемость. Средняя оценка по текущему предмету выводится в последней колонке и делается не редактируемой. Иначе если в этих, колонках, где должен происходить ввод оценок не прошла проверка на действительность, то введенный текст просто замещается пустотой.

Функция выбора предмета `on_comboBox_2_activated` вызывается при изменении значения `comboBox_2`.

```
void Widget::on_comboBox_2_activated(int index)
{
    if (index!=-1) {
        currentSubject=index;
        setupGrades();
    }
}
```

Листинг 16. Функция выпадающего списка при выборе предмета.

Здесь если выбран предмет, то присваивается индекс глобальной переменной `currentSubject`, отвечающей за выбранный предмет, а так же вызвать функцию `setupGrades()`. Проверка на индекс идет, так как при не выбранном предмете ничего не должно происходить. `setupGrades()` нужен, чтобы расставить оценки в таблице.

```
void Widget::setupGrades()
{
    if (currentSubject!=-1) {
        formatTable();
        for (int j=0; j<ui->tableWidget->rowCount();j++){
            for (int i = 2;i<ui->tableWidget->columnCount()-1;i++)
            {
                ui->tableWidget->setItem(j,i,new
                QTableWidgetItem(groups[currentGroup].students[j].grades[current
                Subject][i-2]));
                if (ui->tableWidget->item(j,1)-
                >background()==QBrush(Qt::yellow))
                {
                    if (!ui->tableWidget->item(j,i))
                        ui->tableWidget->setItem(j,i,new
                        QTableWidgetItem);
                    ui->tableWidget->item(j,i)-
                    >setBackground(QBrush(Qt::yellow));
                }
            }
        }
        enableEdit();
    }
}
```

Листинг 17. Вставка оценок и выделение найденной строки по поиску.

Если предмет выбран, то форматируется таблица и выставляются все оценки. В конце включается возможность редактирования ячеек, отведенных под оценки. Фрагмент кода, использующий методы setBackground() и background() задействованы во время использования другой функции.

По нажатию кнопки удаления ученика вызывается функция on_pushButton_2_clicked.

```
void Widget::on_pushButton_2_clicked()
{
    //DELETE VVV
    bool flag = false;
    int once;
    if (ui->pushButton_2->text()=="Подтвердить/отменить")
    {
        int deletesCount=0;
        for (int i = 0; i<ui->tableWidget->rowCount(); i++)
        {
            if (ui->tableWidget->item(i, 0) -
>checkState()==Qt::Checked)
            {
                if (!flag) once = createmsgSignal("Вы
действительно хотите удалить выбранных учеников?");
                if (!flag && once == QMessageBox::Ok) flag
= true;

                if (flag) {
                    for (int j=i-deletesCount; j<ui->tableWidget-
>rowCount()-deletesCount; j++)
                    {
                        if(j!=29)

groups[currentGroup].students[j]=groups[currentGroup].students[j
+1];

                        else

groups[currentGroup].students[j]=Student("");
                    }
                    deletesCount++;
                }
            }
        }
        flag = true;
        ui->tableWidget->hideColumn(0);
        ui->pushButton_2->setText("Удаление учеников");
        if (ui->comboBox_2->currentIndex()==-1) {
            setupTables();
            ui->pushButton->setEnabled(true);
            ui->comboBox->setEnabled(true);
            ui->comboBox_2->setEnabled(true);
        }
    }
}
```

```

        else{
            setupTables();
            setupGrades();
            ui->pushButton->setEnabled(true);
            ui->comboBox->setEnabled(true);
            ui->comboBox_2->setEnabled(true);
        }
    }
    //CHECKBOXES VVV
    if (ui->pushButton_2->text()=="Удаление учеников"&&ui->
    >tableWidget->rowCount()!=0&&!flag){
        ui->tableWidget->showColumn(0);
        ui->pushButton->setEnabled(false);
        ui->comboBox->setEnabled(false);
        ui->comboBox_2->setEnabled(false);
        for (int i = 0; i<ui->tableWidget->rowCount();i++)
        {
            QTableWidgetItem *check = new QTableWidgetItem();
            check->setData(Qt::CheckStateRole,Qt::Unchecked);
            check->setFlags(check->flags() &
            ~Qt::ItemIsEditable);
            ui->tableWidget->setItem(i, 0, check);
        }
        ui->pushButton_2->setText("Подтвердить/отменить");
    }
}

```

Листинг 18. Удаление ученика.

Одна и та же кнопка выполняет две последовательные задачи. Функция определяет, какую из задач сейчас надо выполнять, посредством переименования кнопки. Вначале, если кнопка имеет имя «Удаление учеников», показывается первая колонка (showColumn()) выключаются некоторые конфликтующие элементы формы, а далее первая колонка полностью заполняется чекбоксами. Для этого создается новый объект QTableWidgetItem, ему с помощью метода setData() устанавливается роль Qt::CheckStateRole с состоянием Qt::Unchecked по умолчанию, также он делается неизменяемым, так как объект по прежнему содержит в себе текст и возможность его редактировать. После полного заполнения кнопка меняет название на «Подтвердить/отменить».

Если название кнопки «Подтвердить/отменить», то начинается процесс удаления выбранных учеников. Пройдясь по первой колонке с чекбоксами проверяем на наличие галочки с помощью условия checkState() ==

Qt::Checked. Вначале в функции проверяется с помощью булевой переменной `flag`, было ли выведено сообщение о подтверждении в первый раз. Если нет, то сообщение `QMessageBox` выводится, запоминается индекс нажатой кнопки и флаг меняется. Далее если кнопка была `QMessageBox::Ok`, то происходит смещение массива с удалением. Начиная с индекса ученика, которого надо удалить, происходит замещение текущего ученика следующим. Получается, что пропадает лишь удаляемый ученик, а остальные смещаются в массиве. Переменная `deletescount` отвечает за количество таких удалений, в этом случае общий обход всего массива будет уменьшаться на это количество (так как количество элементов уменьшается). При этом происходит проверка не последний ли элемент удаляется, если так, то смещения не происходит, а ученик просто замещается пустым. В конце первая колонка снова скрывается, кнопка переименовывается обратно в «Удаление ученика» и происходит настройка интерфейса в зависимости от выбранных комбобоксов.

Кнопка «Редактировать» вызывает функцию `on_pushButton_3_clicked`.

```
void Widget::on_pushButton_3_clicked()
{
    if (ui->tableWidget->rowCount() != 0) {
        ui->pushButton->setEnabled(false);
        ui->pushButton_4->setEnabled(false);
        ui->pushButton_5->setEnabled(false);
        ui->comboBox->setEnabled(false);
        ui->comboBox_2->setEnabled(false);

        for (int i = 0; i < ui->tableWidget->rowCount(); i++)
        {
            QTableWidgetItem *check = new QTableWidgetItem();
            check->setData(Qt::CheckStateRole, Qt::Unchecked);
            check->setFlags(check->flags() & ~Qt::ItemIsEditable);
            ui->tableWidget->setItem(i, 0, check);
        }
        ui->tableWidget->showColumn(0);
    }
}
```

Листинг 19. Добавление чекбоксов кнопкой «Редактировать».

В теле самой функции происходит лишь отключение всех конфликтующих элементов интерфейса и добавление чекбоксов в первой колонке, подобно функции удаления.

Основной процесс редактирования происходит в теле функции `on_tableWidget_cellChanged`.

```
//EDIT VVV
    if(ui->tableWidget->item(row, column) -
>checkState()==Qt::Checked && ui->pushButton_2-
>text()!="Подтвердить/отменить")
    {
        //QString text = QInputDialog::getText(0,"Input
dialog","Введите имя заново",QLineEdit::Normal,ui->tableWidget-
>item(row,1)->text());
        ndlg = new NameDialog();

connect(ndlg,SIGNAL(sendData(QString,QString)),this,SLOT(receive
Data(QString,QString)));

connect(this,SIGNAL(sendData(QString,QString)),ndlg,SLOT(receive
Data(QString,QString)));
        emit
sendData(groups[currentGroup].students[row].getName(),groups[cur
rentGroup].students[row].getSurn());
        ndlg->exec();
        if (receivedname != ""&&receivedsurn!="")
        {

groups[currentGroup].students[row].setName(receivedname);

groups[currentGroup].students[row].setSurn(receivedsurn);
        receivedname="";
        receivedsurn="";
        ui->tableWidget->hideColumn(0);
        if (ui->comboBox_2->currentIndex()==-1) {
            setupTables();
        }
        else{
            setupTables();
            setupGrades();
        }
        ui->pushButton->setEnabled(true);
        ui->pushButton_3->setEnabled(true);
        ui->pushButton_4->setEnabled(true);
        ui->pushButton_5->setEnabled(true);
        ui->comboBox->setEnabled(true);
        ui->comboBox_2->setEnabled(true);
    }
}
```

Листинг 20. Изменение ячейки, фрагмент о выборе чекбокса для изменения имени ученика.

Сигнал изменения значения ячейки реагирует на изменение состояния, возвращаемого методом `checkState()`. Оно должно быть равно `Qt::Checked`. Для того, чтобы изменение чекбокса не вызывало эту функции во время удаления учеников, проверяется, не названа ли кнопка `pushButton_2` «Подтвердить/отменить», ведь это название появляется только в момент выбора удаляемых учеников и никогда больше. Здесь сначала вызывается новый `NameDialog`, происходит связь как в случае с добавлением ученика, только добавляется еще одна обратная, чтобы передать текущие значения имени и фамилии в поля вызванной формы. Если значение было получено, то просто заменяются атрибуты имени и фамилии у ученика и происходит перенастройка таблицы и интерфейса (заново включаются все кнопки).

По нажатию кнопки «Поиск» вызывается функция `on_pushButton_4_clicked`.

```
void Widget::on_pushButton_4_clicked()
{
    bool is_global = true;
    int once = 0;
    QString gotname = "";
    if (currentGroup!=-1) {
        once = createmsgSignal("Вы хотите начать поиск
только по выбранному классу?");
        if (once==QMessageBox::Ok)
            is_global = false;
    }
    if (once != QMessageBox::Close && once!=0) {
        QString text = QInputDialog::getText(0, "Input
dialog", "Поиск", QLineEdit::Normal, "");
        if (text != "") {
            if (is_global) {
                ui->tableWidget->clearContents();
                ui->tableWidget->setRowCount(0);
                ui->comboBox->setCurrentIndex(-1);
                ui->comboBox_2->setCurrentIndex(-1);
                currentSubject=-1;
                currentGroup=-1;
                formatTable();
                for (int i = 0; i<grup_num;i++)
                    for (int j = 0;j<stud_num;j++) {
                        gotname = groups[i].students[j].getName();
                        int c=0;
```

```

                                while
(gotname[c].toLowerCase()==text[c].toLowerCase() && gotname.size()>=c &&
text.size()>=c)
    {
        c++;
    }
    if (c>1 && gotname[c-1].toLowerCase()==text[c-
1].toLowerCase())
    {
        ui->tableWidget->setRowCount(ui-
>tableWidget->rowCount()+1);
        ui->tableWidget->setItem(ui-
>tableWidget->rowCount()-1,0,new QTableWidgetItem(QString("%1
Класс").arg(QString::number(i+1))));
        ui->tableWidget->setItem(ui-
>tableWidget->rowCount()-1,1,new
QTableWidgetItem(groups[i].students[j].getSurn()+"
"+groups[i].students[j].getName()));
        ui->tableWidget->item(ui->tableWidget-
>rowCount()-1,1)->setBackground(QBrush(Qt::white));
    }else {
        gotname =
groups[i].students[j].getSurn();
        int c=0;
        while
(gotname[c].toLowerCase()==text[c].toLowerCase() && gotname.size()>=c &&
text.size()>=c)
        {
            c++;
        }
        if (c>1 && gotname[c-
1].toLowerCase()==text[c-1].toLowerCase())
        {
            ui->tableWidget->setRowCount(ui-
>tableWidget->rowCount()+1);
            ui->tableWidget->setItem(ui-
>tableWidget->rowCount()-1,0,new QTableWidgetItem(QString("%1
Класс").arg(QString::number(i+1))));
            ui->tableWidget->setItem(ui-
>tableWidget->rowCount()-1,1,new
QTableWidgetItem(groups[i].students[j].getSurn()+"
"+groups[i].students[j].getName()));
            ui->tableWidget->item(ui-
>tableWidget->rowCount()-1,1)->setBackground(QBrush(Qt::white));
        }
    }
}
ui->tableWidget->showColumn(0);
ui->pushButton->setEnabled(false);
ui->pushButton_2->setEnabled(false);
ui->pushButton_3->setEnabled(false);
ui->pushButton_5->setEnabled(false);
ui->comboBox_2->setEnabled(false);

```

```

        }

        else {
            for (int j = 0; j < ui->tableWidget->
rowCount(); j++) {
                gotname =
groups[currentGroup].students[j].getName();
                int c=0;
                while
(gotname[c].toLowerCase()==text[c].toLowerCase() && gotname.size()>c &&
text.size()>c)
                {
                    c++;
                }
                if (c>1 && gotname[c-1].toLowerCase()==text[c-
1].toLowerCase())
                {
                    for (int i = 1; i < ui->tableWidget->
columnCount(); i++) {
                        if (!ui->tableWidget->item(j,i))
                            ui->tableWidget->setItem(j,i,new
QTableWidgetItem);
                        ui->tableWidget->item(j,i)-
>setBackground(QBrush(Qt::yellow));
                    }
                }else{
                    gotname =
groups[currentGroup].students[j].getSurn();
                    int c=0;
                    while
(gotname[c].toLowerCase()==text[c].toLowerCase() && gotname.size()>c &&
text.size()>c)
                    {
                        c++;
                    }
                    if (c>1 && gotname[c-
1].toLowerCase()==text[c-1].toLowerCase())
                    {
                        for (int i = 1; i < ui->tableWidget->
columnCount(); i++) {
                            if (!ui->tableWidget->
item(j,i))
                                ui->tableWidget->
setItem(j,i,new QTableWidgetItem);
                            ui->tableWidget->item(j,i)-
>setBackground(QBrush(Qt::yellow));
                        }
                    }
                }
            }
        }
    }
}
}
}
}

```

Листинг 21. Поиск.

В поиске поддерживается два режима поиска: глобальный и локальный. Вначале вызывается `QMessageBox` с вопросом делать ли поиск локальным или нет. При нажатии кнопки `QMessageBox::Ok` флаг `is_global` меняется на `false`. Далее вызывается поле для ввода поиска.

Если поиск глобальный, то вся таблица и сохраняемые индексы сбрасываются. Форматируется таблица, оставляя лишь 2 первых колонки. Далее идет поиск по всем классам и ученикам. Сначала забирается имя у ученика. Далее посимвольно происходит проверка на совпадение символов полученного имени и введенного текста, символы каждый раз переводятся в их строчный вариант методом `toLowerCase()`. Каждый раз, когда найдено совпадение увеличивается счетчик. Цикл `while` длится до первого не совпадения или пока не достигнет конца одной из строковых переменных `gotname` (имя ученика) или `text` (введенный текст). Если было найдено хотя бы два совпадения, то происходит добавление строки, имени ученика во 2 колонке, добавление номера класса в 1 колонке, а также происходит перекраска ячейки имени в белый цвет методом `setBackground()`. Где в качестве аргумента выступает конструктор класса кисти `QBrush` с аргументом цвета `Qt::white`. Если поиск по имени не дал результатов, то начинается аналогичный цикл по поиску совпадений в фамилии. В конце поиска показывается первая колонка и выключаются некоторые кнопки.

Если поиск локальный, то процедура поиска совпадений идентична глобальному, однако происходит в текущем выбранном классе. Вместо вывода строки с учеником и классом строки с найденными учениками выделяются желтым. Происходит проход по всем ячейкам строки и, если в данной ячейке не установлен объект `QTableWidgetItem`, то он выставляется, а если есть, то ячейка красится с помощью `setBackground` в цвет `Qt::yellow`. Для поддержания цвета при смене предмета был добавлен фрагмент кода в функцию `setupGrades()`, отвечающий за перекраску строчек. Функция проверяет, перекрашена ли в желтый вторая колонка с именем (поскольку

при смене предмета она не меняется, а при смене класса должен происходить сброс выделения). Если да, то идет цикл перекраски всей строки, аналогичный код в функции локального поиска.

Для работы с поиском была отдельно создана функция `on_tableWidget_cellClicked`, которая вызывается при каждом нажатии на какую-либо ячейку.

```
void Widget::on_tableWidget_cellClicked(int row, int column)
{
    if (row!=-1&&column!=-1) {
        if (ui->tableWidget->item(row,1)-
>background()==QBrush(Qt::yellow))
        {
            for (int i = 1; i<ui->tableWidget-
>columnCount();i++){
                if (!ui->tableWidget->item(row,i))
                    ui->tableWidget->setItem(row,i,new
QTableWidgetItem);
                ui->tableWidget->item(row,i)-
>setBackground(QBrush(Qt::white));
            }
        }
        if (ui->tableWidget->columnCount()==2) {
            QStringList klass=ui->tableWidget->item(row,0)-
>text().split(" ");
            QStringList surn=ui->tableWidget->item(row,1)-
>text().split(" ");
            ui->comboBox->setCurrentIndex(klass[0].toInt()-1);
            currentGroup=klass[0].toInt()-1;
            Widget::on_comboBox_activated(currentGroup);
            for(int j=0;j<ui->tableWidget->rowCount();j++){

                if(groups[currentGroup].students[j].getSurn()==surn[0] &&
groups[currentGroup].students[j].getName()==surn[1]){
                    for (int i = 1; i<ui->tableWidget-
>columnCount();i++){
                        if (!ui->tableWidget->item(j,i))
                            ui->tableWidget->setItem(j,i,new
QTableWidgetItem);
                        ui->tableWidget->item(j,i)-
>setBackground(QBrush(Qt::yellow));
                    }
                    break;
                }
            }
        }
    }
}
```

Листинг 22. Выделение строки или переход в нужный класс при поиске.

Если ряд и колонка не минус единица (исключение артефактов) и, если цвет первой колонки желтый (выделение при локальном поиске), то происходит перекраска всей строки белым. Если количество колонок равно двум (такое количество колонок встречается только при выводе глобального поиска), то создаются QStringList от значений текста первой и второй колонок методом split() по пробелам “ “. Так получают списки отдельных слов, разделенных пробелами в изначальной цельной строке. Первое слово первой колонки – номер класса, поэтому в зависимости от него устанавливается currentGroup. На основании этого вызывается функция on_comboBox_activated() от currentGroup, которая выводит найденный класс в таблицу. В этом классе происходит проход по всем строкам с проверкой фамилии и имени ученика с нулевым и первым элементов QStringList surn, где находятся имя и фамилия из поиска. Если полное совпадение было найдено, то происходит выделение всей строки желтым и остановка цикла.

Кнопка «Новый учебный год» переводит учеников, у которых атрибут isViable == true, в следующий класс, иначе оставляет их в нынешнем классе.

```
void Widget::on_pushButton_5_clicked()
{
    currentGroup=-1;
    ui->comboBox->setCurrentIndex(-1);
    currentSubject=-1;
    ui->comboBox_2->setCurrentIndex(-1);
    ui->pushButton->setEnabled(false);
    if (createmsgSignal("Процедура смены учебного года полностью
автоматическая!\nУченики, у которых нет половины посещений или
минимум среднего балла 2.5 по каждому из предметов, остаются на
второй год.\nВы хотите начать новый учебный
год?")==QMessageBox::Ok) {
        int shift =0;
        int newshift=0;
        int currentshift =0;
        for (int i=grup_num-1;i>=0;i--) {
            shift=newshift;
            newshift=0;
            currentshift=0;
        }
        for (int j=0;j<stud_num;j++) {
            if (i==grup_num-1) {
                if(groups[i].students[j].isViable) {
                    groups[i].students[j] = Student("");
                } else if (groups[i].students[j].getName() != "") {
```



```

groups[i].students[newshift]=Student(groups[i].students[j].getName(),groups[i].students[j].getSurn());
        if (newshift!=j)
            groups[i].students[j]=Student("");
            newshift++;
    }
} else {
    if
(groups[i].students[j].isViable&&groups[i].students[j].getName()
!="") {

groups[i+1].students[currentshift+shift]=Student(groups[i].students[j].getName(),groups[i].students[j].getSurn());
            groups[i].students[j]=Student("");
            currentshift++;
        } else if(groups[i].students[j].getName()!="") {

groups[i].students[newshift]=Student(groups[i].students[j].getName(),groups[i].students[j].getSurn());
            if (newshift!=j)
                groups[i].students[j]=Student("");
                newshift++;
        }
    }
}
}
}
}
ui->tableWidget->clearContents();
ui->tableWidget->setRowCount(0);
}

```

Листинг 23. Функция начала нового учебного года с переводом учеников.

Вначале очищается таблица. Затем выводится QMessageBox с предупреждением о полной автоматизации процесса, так как основывается на автоматически рассчитываемых данных средних баллов middleGrade и посещаемости absence. При подтверждении происходит процесс смещения учеников. Происходит обратный проход по всем классам, в каждом классе идет обход каждого ученика. Для последнего класса идет отдельная последовательность действий, так как прошедшие ученики просто удаляются. Если ученик остается на второй год, то ученик находящийся на позиции newshift (смещение в текущем классе на количество второкурсников) становится учеником, которого сейчас проверяют. Если между местом, куда надо поставить ученика и изначальным местом ученика в списка есть

промежуток, то на изначальное место ставится пустой ученик. При смене класса newshift обнуляется, а shift (смещение в классе, куда переходит ученик) становится равен newshift до этого. Для всех остальных классов, если ученик существует и проходит в следующий класс, на место ученика в следующем классе (который уже изменен, в нем остались только второгодники) на позиции shift+currentshift (смещение в следующем классе на второгодников + смещение по количеству перешедших в него учеников) ставится проверяемый ученик, а в предыдущем классе он удаляется. Если ученик не проходит, то смещение происходит аналогично последнему классу.

Далее нужно вкратце описать реализацию NameDialog. По сути это просто другая форма, работающая по тем же принципам, просто некоторые ее функции связаны с основной формой через сигналы и слоты. Заголовочный файл namedialog.h включает в себя реализацию класса окна приложения. Оно наследует методы встроенного класса QDialog, а также включает в себя объявления всех функций окна (сигналы и слоты), включая конструктор и деструктор, объявления всех пользовательских функций для Namedialog. Здесь же создается экземпляр класса Namedialog во внешней области видимости UI.

Реализация функций класса NameDialog располагается в файле namedialog.cpp.

```
NameDialog::NameDialog(QWidget *parent) :
    QDialog(parent),
    nui(new Ui::NameDialog)
{
    nui->setupUi(this);
}

NameDialog::~NameDialog()
{
    delete nui;
}

void NameDialog::receiveData(QString Name, QString Surn) {
    nui->lineEdit->setText(Name);
}
```

```

        nui->lineEdit_2->setText(Surn);
        nui->label_3->setText("Изменение имени и фамилии.");
    }
    void NameDialog::on_pushButton_clicked()
    {
        emit sendData(nui->lineEdit->text(), nui->lineEdit_2->
text());
        this->close();
    }

    void NameDialog::on_pushButton_2_clicked()
    {
        emit sendData("", "");
        this->close();
    }

```

Листинг 24. Namedialog.cpp.

Здесь аналогично widget.cpp находятся конструктор и деструктор. Здесь находится слот receiveData(), который получив сигнал sendData от Widget выводит полученные данные QLineEdit и меняет текст QLabel на оповещение об изменении (этот слот связывается только при редактировании). Две кнопки pushButton и pushButton_2 в их определениях слотов clicked() отправляют сигнал sendData окну Widget с данными из QLineEdit или же пустые значения при отмене. Слово emit является лишь подсказкой программисту о том, что данная функция – сигнал. Emit в среде определяется лишь как #define emit.

Кроме того, в программе используются некоторые простые или декоративные функции.

```

int Widget::createmsgSignal(QString msg) {
    QMessageBox msg;
    msg.setWindowTitle("Журнал");
    msg.setText("Внимание");
    msg.setInformativeText(msg);
    msg.setStandardButtons(QMessageBox::Ok |
QMessageBox::Cancel);
    msg.setDefaultButton(QMessageBox::Ok);
    return msg.exec();
}
bool Widget::checkGrade(QString Check) {
    if ((Check[0]>49&&Check[0]<54) || Check=="H")
        return true;
    else return false;
}
void Widget::receiveData(QString Name, QString Surn) {

```

```

        if (Name!=""&&Surn!="") {
            receivedname=Name;
            receivedsurn=Surn;
        }
        else {receivedname="";
            receivedsurn="";}
    }
    void Widget::restrictEdit() {
        if(ui->tableWidget->rowCount()>0&&ui->tableWidget->columnCount()>0) {
            for (int i =0;i<ui->tableWidget->rowCount();i++)
                for (int j=2;j<ui->tableWidget->columnCount();j++)
                {
                    if (!ui->tableWidget->item(i,j))
                        ui->tableWidget->setItem(i,j,new
QTableWidgetItem);
                    ui->tableWidget->item(i,j)->setFlags(ui->tableWidget->item(i,j)->flags() & ~Qt::ItemIsEditable);
                }
            }
        }
    void Widget::enableEdit() {
        if(ui->tableWidget->rowCount()>0&&ui->tableWidget->columnCount()>0) {
            for (int i =0;i<ui->tableWidget->rowCount();i++)
                for (int j=2;j<ui->tableWidget->columnCount()-1;j++)
                {
                    if (!ui->tableWidget->item(i,j))
                        ui->tableWidget->setItem(i,j,new
QTableWidgetItem);
                    ui->tableWidget->item(i,j)->setFlags(ui->tableWidget->item(i,j)->flags() | Qt::ItemIsEditable);
                }
            }
        }
    }
}

```

Листинг 25. Малые функции вызова сообщения, проверки, слота и разрешения редактирования ячеек.

createmsgSignal() вызывает QMessageBox по заданному шаблону. checkGrade() проверяет введенный символ на нахождение в интервале [2;5] или букве «н». receiveData() – слот для получения данных из NameDialog. restrictEdit() и enableEdit() – запрещают и разрешают ввод данных в ячейки, где расположены оценки.

3. Тестирование приложения

Тестирование – важный процесс любой программы, помогающий выявлять ошибки и улучшать приложение. Сейчас ни одна серьезная программа не выпускается, пока она не пройдет все этапы тестирования. Нахождение дефектов происходит путем сравнения ожидаемого результата от какого-либо действия с фактическим. Чаще всего ошибки (или по-другому баги) являются отклонением от спецификации. Существует множество видов и направлений тестирования, которые можно делить на классы по большому количеству признаков. В данной работе мы будем использовать ручное тестирование со стороны обычного пользователя, не имеющего доступа к коду (т.е. методом «черного ящика»). Остальные применяемые нами виды тестирования будут указаны в тест-плане – документе, описывающем организацию процесса тестирования.

3.1. Тест-план

1. Объект тестирования: приложение «Школа», реализующее школьный журнал.

2. Виды тестирования и его стратегия.

Применяются следующие виды тестирования:

- Тестирование методом «черного ящика», поскольку код программы небольшой и сам код много раз проверялся в процессе его написания
- Ручное тестирование, так как это просто, не требует навыков написания скриптов, и быстро, нет необходимости писать какой-либо код. В рамках нашей небольшой программы быстрее будет все проверить вручную.
- Позитивное и негативное тестирование. Необходимо для понимания работы приложения не только в «ожидаемых» условиях, но и при неумелом использовании программы.
- Функциональное тестирование, поскольку работоспособность функционала приложения является одним из главных критериев его оценки пользователями.

- Тестирование интерфейса пользователя. Интерфейс – это то, на что пользователь смотрит в первую очередь. Важно, чтобы он был удобен и, главное, корректно работал. Более того, проверка интерфейса необходима, поскольку это наш первый проект создания приложения с графическим интерфейсом на Qt, из-за чего могут возникнуть непредвиденные ошибки.
- Альфа-тестирование, поскольку программа создается в первый раз. Не планируется, что им будут пользоваться большое количество пользователей, поэтому бета-тестирование и остальные виды тестирования по его времени проведению слишком дороги и не оправдывают затраченных на них усилий.
- Тестирование по документации. Объем требуемой документации небольшой, срок сдачи приложения длинный, поэтому для улучшения качества тестирования принято делать его по документации.
- Динамическое тестирование – т.е. с запуском приложения. Так как ключевой задачей тестирования является проверка реального поведения приложения.

Не будут применяться следующие виды тестирования:

- Тестирование локализации, поскольку приложение пишется для пользователей из России (язык интерфейса – русский, пятибалльная система оценивания).
- Тестирование производительности. Программа не требует доступа к интернету и предполагает использование лишь одним пользователем. Изначально в программе предполагается не более 330 (11 классов, в каждом максимум 30 человек) записей, что является совсем незначительным объемом информации для современных приложений.
- Статическое тестирование, так как программа не подразумевает от пользователя работы с какими-либо посторонними файлами, кроме самого приложения, и изменения настроек среды исполнения приложения.
- Регрессионное тестирование, поскольку создается только первая версия приложения, предыдущих версий не существует.

Стратегия: принято решение вести чек-листы и писать тест-кейсы, чтобы не упустить какую-либо часть функционала, так же, чтобы не запутаться будет создана карта приложения, для написания тест-кейсов будет использоваться Excel, приоритетом тестирования является необходимый функционал программы (возможность добавления/удаления новых учеников, учета оценок и посещаемости).

3. Критерии начала тестирования: готовность спецификации и разработанного кода приложения.

4. Критерии окончания тестирования: окончание срока сдачи приложения, а также количество пройденных тест-кейсов не менее 20.

3.2. Карта приложения

Детальная карта приложения является хорошей базой к созданию чек-листов, а затем тест-кейсов. Графическое представление всех необходимых для проверки частей приложения помогает не забыть о какой-либо функции. Для создания карты мы будем пользоваться онлайн-программой для создания интеллект-карт MindMeister.

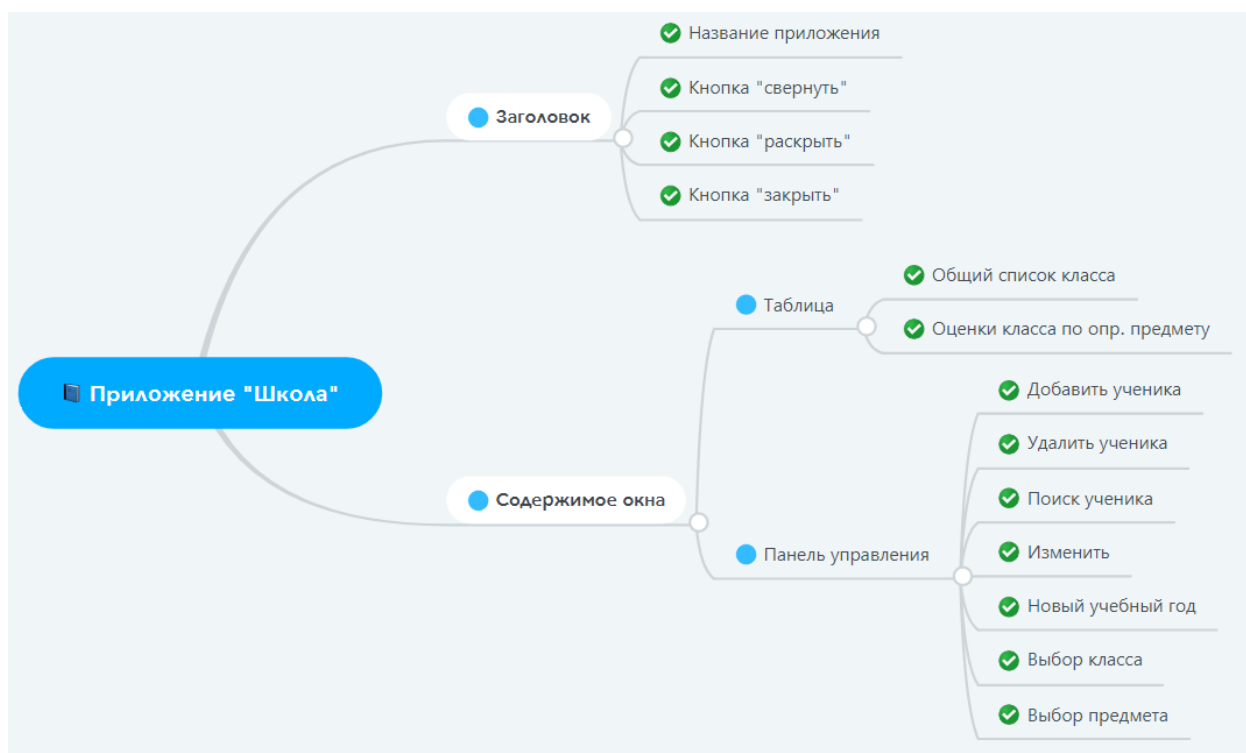


Рис. 7. Карта приложения.

Зелеными знаками с галочками помечается, что конкретно необходимо проверить.

3.3. Создание чек-листов

На основе карты приложения и спецификации легко можно создать чек-листы – перечень шагов или функциональности, необходимые для проверки корректности работы приложения. Шаги должны быть лаконично сформулированы и не должны повторяться. На основе карты приложения получаются вот такие чек-листы:

1. Проверка заголовка.

1.1. Проверить название приложения.

1.2. Проверить работу кнопки «свернуть».

1.3. Проверить разворачивание приложения из панели задач.

1.4. Проверить работу кнопки «развернуть».

1.5. Проверить работу кнопки «закрыть».

2. Проверка содержимого окна.

2.1. Проверка таблицы.

2.1.1. Убедиться, что для каждого класса конкретно отображается список учащихся и оценок.

2.1.2. Проверить, чтобы нельзя было изменять значения ячеек, в которых хранятся значения о средних оценках и посещаемости.

2.1.3. Убедиться, что для каждого класса конкретно отображается журнал его оценок по конкретному предмету.

2.1.4. Проверить, чтобы нельзя было через ячейку изменить имя и фамилию ученика.

2.1.5. Проверить, чтобы в ячейки с оценками нельзя было вводить буквы (кроме «н»), символы, числа, не являющиеся оценками от 2 до 5.

2.2. Проверка панели управления.

2.2.1. Убедиться, что можно добавлять ученика, только в определенный класс.

2.2.2. Убедиться, что ученик добавляется корректно. Совпадают введенные имя и фамилия с теми, что в таблице.

2.2.3. Проверить, что при добавлении больше 30 учеников в один класс, приложение продолжает работать.

2.2.4. Проверить удаление учеников.

2.2.5. Проверить поиск по имени, фамилии в конкретном классе и во всех классах.

2.2.6. Убедиться, что при изменении ученика, корректно меняются его имя, фамилия.

2.2.7. Проверить, чтобы при начале следующего учебного года, все ученики с достаточной посещаемостью и без долгов переводились в следующий класс. Одиннадцатиклассники должны удаляться.

2.2.8. Убедиться, что при выборе класса выбор предметов соответствует программе этого класса.

2.2.9. Убедиться, что при выборе класса или класса и предмета корректно отображается список учащихся.

3.4. Создание тест-кейсов.

Тест-кейсы — это подробная реализация каждого пункта чек-листа, если он есть, состоящая из набора входных данных, условий выполнения и ожидаемых результатов. Задача любого тест-кейса проверка поведения программы. Также тест-кейсы можно создавать и без чек-листа. Результатом тест-кейса является совпадение или несовпадение ожидаемого результата с фактическим. Тест-кейс считается успешно пройденным только при совпадении ожидаемого результата с фактическим на каждом шагу.

Тест-кейсы для нашего приложения:

id	Заголовок	Предусловие	Шаги	Ожидаемый результат	Результат
1	Название приложения		Открыть приложение	В левом углу заголовка программы написано название "Журналы"	Успешен

2	Проверка кнопки "свернуть"	Открыть приложение	Нажать на кнопку "свернуть"	Приложение свернулось, на панели задач отображается иконка.	Успешен
			Нажать на иконку приложения на панели задач	Приложение открылось.	
3	Проверка кнопки "развернуть"	Открыть приложение	Нажать на кнопку "развернуть"	Приложение развернулось на весь экран.	Успешен
			Нажать на кнопку "развернуть"	Приложение свернулось в прежний формат	
4	Проверка кнопки "закрыть"	Открыть приложение	Нажать на кнопку "закрыть"	Приложение закрылось	Успешен
5	Список классов	Открыть приложение	Нажать на кнопку под текстом "Класс"	Выпадет список классов от 1 до 11.	Успешен
			Выбрать 1 класс	В верхних столбцах таблицы появится текст "Ср. оценка" и название предмета, изучаемого в 1 классе. В последнем столбце "Посещаемость"	
			Повторить предыдущие два пункта, пока не будут проверены все классы	В верхних столбцах таблицы появится текст "Ср. оценка" и название предмета, изучаемого в выбранном классе. В последнем столбце "Посещаемость"	
6	Оценки выбранного класса по опр. предмету	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс	Нажать на кнопку под текстом "Предмет"	Выпадет список предметов, изучаемых в выбранном классе	Успешен
			Выбрать первый предмет	Столбцы после имени будут пронумерованы от 1 до 14, затем в последнем столбце "Ср. оценка"	
7	Добавление ученика	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс	Нажать кнопку "Новый ученик"	Появится окно, где будет написано "Введите имя и фамилию нового ученика", под надписью 2 строки ввода данных (отдельно) для имени и фамилии, затем две кнопки "ок" и "отмена".	Успешен
			Ввести значение в поля "Имя", состоящее из любых символов	В поле "Имя" отображается введенный текст	
			Ввести значение в поля "Фамилия", состоящее из любых символов	В поле "Фамилия" отображается введенный текст	

			Нажать "Ок"	Имя и фамилия нового ученика записана во втором левом столбце	
8	Отмена добавления ученика	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс	Нажать кнопку "Новый ученик"	Появится диалоговое окно, где будет написано "Введите имя и фамилию нового ученика", под надписью 2 строки ввода данных (отдельно) для имени и фамилии, затем две кнопки "ок" и "отмена"	Успешен
			Ввести значение в поля "Имя", состоящее из любых символов	В поле "Имя" отображается введенный текст	
			Ввести значение в поля "Фамилия", состоящее из любых символов	В поле "Фамилия" отображается введенный текст	
			Нажать "Отмена"	На форме ничего не изменилось. Диалоговое окно закрылось	
9	Удаление 1 ученика	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать двух новых учеников	Нажать на кнопку "Удаление учеников"	Во втором столбце около каждого ученика появился квадратик	Успешен
			Нажать на квадратик около первого ученика	В квадратике появилась галочка	
			Нажать на кнопку "Подтвердить/отменить"	Появится всплывающее окно "действительно ли вы хотите удалить выбранных учеников?"	
			Нажать "Ок"	Запись о ученике из таблицы исчезла. Второй ученик поднялся на строку вверх	
10	Удаление нескольких учеников	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать минимум 5 учеников	Нажать на кнопку "Удаление учеников"	Во втором столбце около каждого ученика появился квадратик	Успешен
			Нажать на квадратики около нескольких учеников	В выбранных квадратах появились галочки	
			Нажать на кнопку "Подтвердить/отменить"	Появится всплывающее окно "действительно ли вы хотите удалить выбранных учеников?"	
			Нажать "Ок"	Запись о выбранных учениках из таблицы исчезла. Список учеников сдвинулся вверх.	
11	Отмена удаления 1	Открыть приложение,	Нажать на кнопку "Удаление учеников"	Во втором столбце около ученика появилась галочка	Успешен

	ученика	нажать на кнопку под текстом "Класс", выбрать любой класс, создать 1 ученика	Нажать на квадратик около ученика	В выбранном квадратики появилась галочка	
			Нажать на кнопку "Подтвердить/отменить"	Появится всплывающее окно "действительно ли вы хотите удалить выбранных учеников?"	
			Нажать "cancel"	Исчезло всплывающее окно	
12	Поиск по существующему имени/фамилии по одному классу	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать трех новых учеников	Нажать на кнопку "Поиск"	Появится диалоговое окно с текстом "Внимание. Вы хотите начать поиск только по выбранному классу?" и кнопками "ок" и "cancel"	Успешен
			Нажать "Ок"	Появится диалоговое окно с текстом "поиск", строкой для ввода имени/фамилии и кнопками "ок" и "cancel"	
			Введите фамилию ученика, существующего в выбранном классе.	Диалоговое окно закрылось. Вся строка, на которой размещена информация об этом ученике, стала желтой	
13	Поиск по не существующему имени/фамилии по одному классу	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать трех новых учеников	Нажать на кнопку "Поиск"	Появится диалоговое окно с текстом "Внимание. Вы хотите начать поиск только по выбранному классу?" и кнопками "ок" и "cancel"	Успешен
			Нажать "Ок"	Появится диалоговое окно с текстом "поиск", строкой для ввода имени/фамилии и кнопками "ок" и "cancel"	
			Ввести имя ученика, не существующего в выбранном классе.	Диалоговое окно закрылось.	
14	Поиск по не существующему имени/фамилии по всей школе	Открыть приложение, Создать в минимум 3 классах минимум по 4 ученика.	Нажать на кнопку "Поиск"	Появится диалоговое окно с текстом "Внимание. Вы хотите начать поиск только по выбранному классу?" и кнопками "ок" и "cancel"	Успешен
			Нажать "cancel"	Появится диалоговое окно с текстом "поиск", строкой для ввода имени/фамилии и кнопками "ок" и "cancel"	
			Ввести фамилию ученика, несуществующего в школе	Диалоговое окно закрылось. Таблица стала пустой, класс и предмет (если был выбран) сбросились.	
15	Поиск по имени/фамилии с ошибкой в 1 букву по одному	Открыть приложение, нажать на кнопку под текстом "Класс",	Нажать на кнопку "Поиск"	Появится диалоговое окно с текстом "Внимание. Вы хотите начать поиск только по выбранному классу?" и кнопками "ок" и "cancel"	Успешен

	классу	выбрать любой класс, создать трех новых учеников	Нажать "Ок"	Появится диалоговое окно с текстом "поиск", строкой для ввода имени/фамилии и кнопками "ок" и "cancel"	
			Ввести фамилию ученика, существующего в выбранном классе, с 1 ошибкой.	Диалоговое окно закрылось. Вся строка, на которой размещена информация об этом ученике, стала желтой	
16	Поиск по существующей фамилии по всей школе	Открыть приложение, создать в минимум 3 классах минимум по 4 ученика.	Нажать на кнопку "Поиск"	Появится диалоговое окно с текстом "Внимание. Вы хотите начать поиск только по выбранному классу?" и кнопками "ок" и "cancel"	Успешен
			Нажать "cancel"	Появится диалоговое окно с текстом "поиск", строкой для ввода имени/фамилии и кнопками "ок" и "cancel"	
			Ввести фамилию ученика, существующего в школе	Диалоговое окно закрылось. В таблице появилось 3 столбца, в 1 столбце номер строки, в 2 столбце класс, найденного ученика, в 3 столбце имя и фамилия найденного ученика.	
17	Закрытие окна добавления ученика через "крестик"	Открыть приложение, выбрать любой класс, создать в нем любого ученика.	Нажать кнопку "Новый ученик"	Появится диалоговое окно, где будет написано "Введите имя и фамилию нового ученика", под надписью 2 строки ввода данных (отдельно) для имени и фамилии)	Провален
			Нажать на крестик в правом верхнем углу диалогового окна	На форме ничего не изменилось. Диалоговое окно закрылось	
18	Перевод на следующий год при отсутствии учеников	Открыть приложение	Нажать кнопку "Начало нового учебного года"	Появится диалоговое окно с предупреждением "Процедура смены учебного года полностью автоматическая. Ученики, у которых нет половины посещений или минимум среднего балла 2.5 по каждому из предметов, остаются на второй год. Вы хотите начать новый учебный год?" и кнопками "ок" и "cancel"	Успешен
			Нажать "cancel"	Диалоговое окно закрылось, ничего не произошло.	

19	Отмена перевода на следующий учебный год при отсутствии учеников	Открыть приложение	Нажать кнопку "Начало нового учебного года "	Появится диалоговое окно с предупреждением "Процедура смены учебного года полностью автоматическая. Ученики, у которых нет половины посещений или минимум среднего балла 2.5 по каждому из предметов, остаются на второй год. Вы хотите начать новый учебный год?" и кнопками "ок" и "cancel"	Успешен
			Нажать "Ok"	Диалоговое окно закрылось, ничего не произошло.	
20	Перевод на следующий год при наличии учеников	Открыть приложение, создать во всех классах по несколько учеников. Сделать несколько учеников с плохой посещаемостью (написать н во все ячейки по какому-нибудь предмету) и неуспеваемостью (средним баллом < 2,5)	Нажать кнопку "Начало нового учебного года "	Появится диалоговое окно с предупреждением "Процедура смены учебного года полностью автоматическая. Ученики, у которых нет половины посещений или минимум среднего балла 2.5 по каждому из предметов, остаются на второй год. Вы хотите начать новый учебный год?" и кнопками "ок" и "cancel"	Успешен
			Нажать "Ok"	Диалоговое окно закрылось. В открытом классе появились ученики, перешедшие в следующий класс. Ученики с плохой успеваемостью или посещаемостью остались в тех же классах, где и были. Кнопка "добавить ученика" стала неактивной (более темного цвета, при нажатии не работает)	
21	Заполнение оценок	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать ученика	Нажать на кнопку под текстом "Предмет"	Выпадет список предметов, изучаемых в выбранном классе	Успешен
			Выбрать любой предмет	Столбцы после имени будут пронумерованы от 1 до 14, затем в последнем столбце "Ср. оценка"	
			Нажать на любую ячейку 1-14 в строке ученика	Ячейка выделится синим цветом	
			Ввести оценку 2-5 и нажать enter	В ячейке появится введенное значение и она выделится синим цветом	
			Аналогичным образом ввести еще несколько оценок	В последнем столбце "Ср. оценка" считается ср. оценка ученика по выбранному предмету как частное суммы оценок на количество всех оценок	

22	Заполнение пропусков	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать ученика, выбрать любой предмет	Нажать на любую ячейку 1-14 в строке ученика	Ячейка выделится синим цветом	Успешен
			Ввести n и нажать enter	В ячейке появится введённое значение и она выделится синим цветом	
23	Изменение средней оценки в таблице	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать минимум 1 ученика	Нажать на любую ячейку "Ср. оценка предмет"	Ячейка выделится синим цветом	Провален
			Ввести любые символы, буквы, числа и нажать enter	В выбранной ячейке ср. оценка не изменилась	
24	Некорректное изменение оценки за урок или н в таблицу	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать минимум 1 ученика, выбрать любой предмет	Нажать на любую ячейку 1-14 в строке ученика	Ячейка выделится синим цветом	Успешен
			Ввести букву, слово, любой символ или числа вне отрезка [2,5] и нажать enter	Ячейка станет пустой и она выделится синим цветом	
25	Добавление больше 30 учеников в один класс	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать в нем 30 учеников	Нажать кнопку "Новый ученик"	Появится диалоговое окно, где будет написано "Введите имя и фамилию нового ученика", под надписью 2 строки ввода данных (отдельно) для имени и фамилии)	Провален
			Ввести значение в поля "Имя", состоящее из любых символов	В поле "Имя" отображается введённый текст	
			Ввести значение в поля "Фамилия", состоящее из любых символов	В поле "Фамилия" отображается введённый текст	
			Нажать "Ок"	Диалоговое окно закрылось. Ничего не произошло, так как таблица переполнена.	
26	Отмена удаления нескольких учеников	Открыть приложение, нажать на кнопку под текстом "Класс",	Нажать на кнопку "Удаление учеников"	Во втором столбце около каждого ученика появился квадратик	Провален

		выбрать любой класс, создать в нем не меньше 3 учеников	Нажать на квадратики около 2 учеников	В выбранных квадратах появились галочки	
			Нажать на кнопку "Подтвердить/отменить"	Появится всплывающее окно "действительно ли вы хотите удалить выбранных учеников?"	
			Нажать "cancel"	Исчезло всплывающее окно	
27	Отмена перевода на следующий учебный год при наличии учеников	Открыть приложение, создать во всех классах по несколько учеников.	Нажать кнопку "Начало нового учебного года"	Появится диалоговое окно с предупреждением "Процедура смены учебного года полностью автоматическая. Ученики, у которых нет половины посещений или минимум среднего балла 2.5 по каждому из предметов, остаются на второй год. Вы хотите начать новый учебный год?" и кнопками "ок" и "cancel"	Успешен
			Нажать "cancel"	Диалоговое окно закрылось, значение выбранного класса и предмета сбрасываются, исчезает заполненная в таблице информация об учениках	
28	Отмена перевода на следующий учебный год при наличии учеников "крестиком"	Открыть приложение, создать во всех классах по несколько учеников.	Нажать кнопку "Начало нового учебного года"	Появится диалоговое окно с предупреждением "Процедура смены учебного года полностью автоматическая. Ученики, у которых нет половины посещений или минимум среднего балла 2.5 по каждому из предметов, остаются на второй год. Вы хотите начать новый учебный год?" и кнопками "ок" и "cancel"	Успешен
			Нажать на крестик в правом верхнем углу диалогового окна	Диалоговое окно закрылось, значение выбранного класса и предмета сбрасываются, исчезает заполненная в таблице информация об учениках	
29	Редактирование имени ученика	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать в нем не	Нажать на кнопку "Редактировать имя"	Во втором столбце около каждого ученика появился квадратик	Успешен

		меньше 3 учеников	Нажать на квадратик около любого ученика	В квадратике появилась галочка. Появится окно, где будет написано "Изменение имени и фамилии", под надписью 2 строки ввода данных (отдельно) для имени и фамилии, затем две кнопки "ок" и "отмена". На форме с таблицей стали неактивными кнопки "класс", "новый ученик", "предмет", "поиск"	
			Ввести значение в поля "Имя", состоящее из любых символов	В поле "Имя" отображается введенный текст	
			Ввести значение в поля "Фамилия", состоящее из любых символов	В поле "Фамилия" отображается введенный текст	
			Нажать "Ок"	В таблице имя и фамилия выбранного ученика изменились на введенные. Окно изменения имени и фамилии закрылось.	
30	Отмена редактирования имени ученика	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать в нем не меньше 3 учеников	Нажать на кнопку "Редактировать имя"	Во втором столбце около каждого ученика появился квадратик	Провален
			Нажать на квадратик около любого ученика	В квадратике появилась галочка. Появится окно, где будет написано "Изменение имени и фамилии", под надписью 2 строки ввода данных (отдельно) для имени и фамилии, затем две кнопки "ок" и "отмена"	
			Нажать "Отмена"	Окно изменения имени и фамилии закрылось. Квадратики слева от учеников исчезли. Стали активными кнопки "класс", "новый ученик", "предмет", "поиск"	
31	Отмена поиска ученика через "крестик"	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс, создать в нем 1 ученика	Нажать на кнопку "Поиск"	Появится диалоговое окно с текстом "Внимание. Вы хотите начать поиск только по выбранному классу?" и кнопками "ок" и "cancel"	Провален
			Нажать на крестик в правом верхнем углу диалогового окна	На форме ничего не изменилось. Диалоговое окно закрылось	
32	Добавление ученика без имени	Открыть приложение, нажать на кнопку под	Нажать кнопку "Новый ученик"	Появится окно, где будет написано "Введите имя и фамилию нового ученика", под надписью 2 строки	Успешен

		текстом "Класс", выбрать любой класс		ввода данных (отдельно) для имени и фамилии), затем две кнопки "ок" и "отмена".	
			Ввести значение в поля "Фамилия", состоящее из любых символов. Поле «Имя» оставить пустым».	В поле "Фамилия" отображается введенный текст	
			Нажать "Ок"	Окно закрылось, в таблице ничего не изменилось, запись о новом ученике добавлена не была	
33	Добавление ученика без фамилии	Открыть приложение, нажать на кнопку под текстом "Класс", выбрать любой класс	Нажать кнопку "Новый ученик"	Появится окно, где будет написано "Введите имя и фамилию нового ученика", под надписью 2 строки ввода данных (отдельно) для имени и фамилии), затем две кнопки "ок" и "отмена"	Успешен
			Ввести значение в поля "Имя", состоящее из любых символов. Поле «Фамилия» оставить пустым».	В поле "Имя" отображается введенный текст	
			Нажать "Ок"	Окно закрылось, в таблице ничего не изменилось, запись о новом ученике добавлена не была	

3.5. Определение дефектов

Отклонение фактического результата от ожидаемого называется багом или дефектом. Описание дефектов похоже на создание тест-кейсов, но отличается тем, что содержит фактический результат. Ожидаемый результат при описании дефекта описывается не для каждого шага, а лишь для всей проверяемой области. Оформление дефектов помогает программистам разобраться в причинах возникновения ошибки и позволяют им самостоятельно не проделывать тест-кейс, руководствуясь лишь информацией из описанного бага.

Из проведенных в предыдущей главе тест-кейсов было провалено 6 тест-кейсов. Для того, чтобы облегчить работу исправления ошибок, нужно оформить дефекты, возникающие при выполнении этих тест-кейсах.

Дефекты:

id деф-та	id связанного тест-кейса	Описание	Предусловие	Шаги воспроизведения	Ожидаемый результат	Фактический результат
1	17	При отмене удаления через крестик создается дубликат	В приложении есть созданный ученик	Нажать кнопку "Новый ученик"	На форме ничего не изменилось. Диалоговое окно закрылось	В выбранном классе создается ученик с таким же именем и фамилии, как последний созданный ученик
				Нажать на крестик в правом верхнем углу диалогового окна		
2	23	Возможность изменения ср. арифм.	В приложении есть созданный ученик, открыт класс с этим учеником	Нажать на ячейку со значением ср. арифм. по любому предмету	Введенное значение не записалось в ячейку. Ср. значение осталось таким же, как и до этого	Введенное значение появилось в ячейке
				Ввести в ячейку букву или цифру и нажать enter		
3	25	Ошибка при добавлении больше 30 учеников в один класс	В выбранном классе создано 30 учеников	Нажать кнопку "Новый ученик"	Выводится сообщение об ошибке, новый ученик не создается	Программа экстренно выключается
				Ввести имя и фамилию ученика		
				Нажать "Ok"		
4	26	Повторное открытие окна - предупреждения об удалении ученика при его закрытии	В выбранном классе не менее 3 учеников	Нажать на кнопку "Удаление учеников"	Исчезло всплывающее окно "Действительно ли вы хотите удалить выбранных учеников?"	Всплывающее окно закрывается и моментально появляется еще раз
				Нажать на квадратик около 2 учеников		
				Нажать на кнопку "Подтвердить/отменить"		
				Нажать "cancel"		
5	30	Невозможность отменить редактирование имени и фамилии	В выбранном классе не менее 1 ученика	Нажать на кнопку "Редактировать имя"	Окно изменения имени и фамилии закрылось. Квадратики слева от учеников исчезли. Стали активными кнопки "Класс", "Новый ученик", "Предмет",	Программа не вышла из режима редактирования. Квадратики не исчезли. Кнопки "Класс", "Новый ученик", "Предмет", "Поиск", "Начало нового учебного
				Нажать на квадратик около любого ученика		

				Нажать "Отмена"	"Поиск", "Начало нового учебного года"	года" неактивны.
6	31	При закрытии поиска поиск все равно происходит	В выбранном классе не менее 1 ученика	Нажать на кнопку "Поиск"	На форме ничего не изменилось. Диалоговое окно закрылось	Появляется диалоговое окно с текстом "поиск", строкой для ввода имени/фамилии и кнопками "ок" и "cancel"
				Нажать на крестик в правом верхнем углу диалогового окна		

3.6. Результаты тестирования

В результате тестирования было обнаружено 6 ошибок в приложении. Используя информацию о дефектах, мы смогли исправить часть багов. Отчет по проделанной работе можно представить в виде таблицы:

id дефекта	Описание	Состояние
1	При отмене удаления через крестик создается дубликат	Исправлено
2	Возможность изменения ср. арифм.	Частично исправлено, значение нельзя менять, но в изначально пустую ячейку до сих пор можно писать
3	Ошибка при добавлении больше 30 учеников в один класс	Исправлено с добавлением предупреждающего сообщения
4	Повторное открытие окна-предупреждения об удалении ученика при его закрытии	Исправлено
5	Невозможность отменить редактирование имени и фамилии	Не будет исправлено, ошибка не нарушает целостность данных
6	При закрытии поиска поиск все равно происходит	Не будет исправлено

Таким образом, тестирование помогло нам улучшить наше приложение, устранить часть дефектов и понять, над чем необходима доработка.

Заключение

В результате нашей курсовой работы мы создали приложение «Школа», поддерживающее режимы учета классов и учеников в них, добавление и удаление учеников, учет посещаемости занятий и оценок учащихся, функцию начала следующего учебного года. Для достижения этой цели было проделано следующее:

- изучена учебная литература по программированию
- произведен анализ предметной области
- построена модель программного обеспечения
- описаны классы, методы и алгоритмы взаимодействия
- разработана структурная схема интерфейса
- разработаны классы
- разработан интерфейс приложения
- составлен тест-план для организации тестирования
- составлена карта приложения
- разработаны чек-листы
- разработаны тест-кейсы
- обнаружены дефекты в изначальной версии приложения
- часть дефектов было устранено
- проанализирован результат проведения тестирования.

Таким образом, мы написали работающее приложение «Школа», выполняющее все требования, поставленные перед ним в спецификации.

Однако применение программы в реальной жизни невозможно, поскольку информация вся информация об учениках исчезает после закрытия приложения и не восстанавливается при его следующем запуске. Поэтому ключевой идеей доработки программы является организация хранения данных в файл и чтения из него. Также необходимо исправить найденные дефекты, которые не удалось устранить при написании курсовой работы.

Помимо этого, для усовершенствования разрабатываемого приложения можно предложить следующее:

- Добавить возможность добавления новых, параллельных школьных классов, которые будут обозначаться буквой. Например, «1 1 А», «2Б» и т.д.
- Добавить возможность добавления новых школьных предметов
- Добавить разделение на дни недели, месяцы
- Добавить автоматическое завершение года при достижении определённого дня
- Добавить возможность ставить несколько оценок в один день

Список используемой литературы

1. Положение о курсовом проектировании, утв. приказом от 11.11.2021 № 398.
2. Тузовский, А.Ф. Объектно-ориентированное программирование: учебное пособие для вузов / А.Ф. Тузовский. — Москва: Издательство Юрайт, 2022. — 206 с. — (Высшее образование). — ISBN 978-5-534-00849-4. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/490369>.
3. Шлее М. Qt 5.10. Профессиональное программирование на C++ (2018) / М. Шлее. — СПб: «БХВ-Петербург», 2018. — 1034 с. — [Электронный ресурс] — URL: https://codernet.ru/books/c_plus/professionalnoe_programmirovanie_na_c_m_shlee/ (дата обращения 11.05.22).