Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

ОТЧЕТ О ПРАКТИКЕ

Кафедра ВТ ИКИТ СФУ

Ознакомительная практика

Руководитель от университета _____ К. В. Пушкарев
подпись, дата

Студент КИ19-09б, 031939831 _____ Е. В. Федченко
подпись, дата

Красноярск 2020

Содержание

1. Тексты заданий

1.1. Текст задания A0

Доработать программу Toynote следующим образом:

Таблица №1 — задания A0

№	Пункт задания	Примечания
1	Добавить пункт «Visit the SibFU Website» в меню Help, открывающий URL http://www.sfu-kras.ru	Открыть URL можно с помощью метода QDesktopServices::openUrl()
2	Добавить свои имя, фамилию, отчество, номер группы и зачётной книжки ниже имени первого автора в диалог «Help | About Toynote...»	
3	Добавить в меню «File» пункт «Exit» для выхода из программы. При активации этого действия, программа должна корректно завершаться	Корректно завершить программу следует вызовом метода QCoreApplication::quit(). Получить указатель на объект QCoreApplication можно через статический метод QCoreApplication::instance()
4	Добавить запрос подтверждения при удалении заметки	Класс QMessageBox позволяет создать окно подтверждения
5	Добавить в меню «File» пункт «Save As Text...», выводящий окно выбора файла и сохраняющий в этот файл заметки в произвольной текстовой форме. Перед каждой заметкой должна выводиться отдельная строка «>>> N/M >>>», после — «<<< N/M <<<», где N — номер заметки (счёт с 1), M — количество заметок. Должны сохраняться все атрибуты заметки, в том числе добавленные вами в последующих заданиях, если такие были	

6	Назначить сочетания клавиш для всех действий, доступных из меню «File»	Назначить сочетания клавиш можно в свойствах действий в Qt Designer
7	Добавить меню «Edit» (пустое)	Добавить меню можно в Qt Designer
8	Добавить пункт «Lottery» в меню «Help», при выборе которого должно открываться однокнопочное окно сообщения с текущей датой и результатами лотереи. В лотерее участвует 20 билетов, из которых N + 1 выигрышных, где N — последняя цифра номера вашей зачётной книжки. Каждый раз, когда пользователь запускает лотерею, случайным образом выбирается один билет и программа сообщает пользователю результат. Если пользователь выиграл, программа сообщает, какой приз он получает. Число различных призов равно числу выигрышных билетов, причём различие не должно быть только количественным (например, призы в 100 рублей и 200 рублей не считаются различными). Призы придумать самостоятельно	Окно сообщения можно отобразить с помощью класса QMessageBox. Текущую дату можно получить с помощью метода QDate::currentDate(). Для генерации случайных чисел использовать генератор типа QRandomGenerator, указатель на которой возвращает метод QRandomGenerator::global()

1.2. Текст задания Б0

Доработать программу А0 следующим образом:

Таблица №2 — задания Б0

№	Пункт задания	Примечания
1	Запретить создавать заметки с пустым заголовком или текстом: выводить окно с сообщением об ошибке и не закрывать диалог	См. класс EditNoteDialog в Toynote. Сообщение об ошибке можно реализовать с помощью QMessageBox
2	Добавить редактирование заметки.	Что должен сделать пользователь

	Должна быть возможность редактировать все устанавливаемые пользователем атрибуты заметки. Устанавливаемые автоматически атрибуты, например дата создания, должны отображаться без возможности изменения. При активации заметки в списке должно открываться окно редактирования (рекомендуется использовать то же окно, что и для добавления). При подтверждении изменений (кнопка «OK»), окно закрывается, и заметка изменяется. При отказе от изменений (кнопка «Cancel»), окно закрывается, и изменения не вносятся. При попытке сохранить заметку с пустым заголовком или текстом выводится сообщение об ошибке, окно редактирования не закрывается, изменения не вносятся	для активации заметки, зависит от параметров его системы. Как правило, это двойной щелчок левой кнопкой мыши или нажатие клавиши Enter. В любом случае об активации заметки сообщает сигнал QAbstractItemView::activated(), который необходимо использовать в этом задании. Список заметок в главном окне является объектом класса QTableView, потомка QAbstractItemView, поэтому у него тоже есть этот сигнал. Список заметок доступен в классе MainWindow по указателю mUi->notesView. Рекомендуется добавить метод в класс Notebook, параметрами которого являются индекс заметки и значение, которое ей необходимо присвоить. Этот метод в конце должен подавать сигнал QAbstractItemModel::dataChanged(), чтобы уведомить привязанные виды об изменении данных

1.3. Текст задания B0

Доработать программу Б0 следующим образом:

Таблица №3 — задания B0

№	Пункт задания	Примечания
1	Добавить отключение списка заметок в главном окне и не имеющих смысла действий, таких как добавление заметки, если записная книжка не открыта.	За отключение виджета (элемента графического интерфейса) отвечает атрибут QWidget::enabled. Меняя этот атрибут посредством метода setEnabled(), который также

	Включать их при открытии записной книжки	является слотом, можно включать и отключать элементы интерфейса. Действия (QAction) имеют аналогичный атрибут — QAction::enabled, а также метод QAction::setEnabled(). Если отключить действие, автоматически отключатся принадлежащие ему виджеты (кнопки на панелях инструментов и пункты меню)

1.4. Текст задания B1

Доработать программу B0 следующим образом:

Таблица №3 — задания B1

№	Пункт задания	Примечания
1	Добавить иконки заметок. В окна создания и редактирования заметки добавить радиокнопки (QRadioButton), позволяющие выбрать одну из трёх иконок для заметки. У каждой кнопки должна отображаться соответствующая иконка (см. QAbstractButton::icon). Выбранная иконка должна отображаться в списке заметок в главном окне. Иконки подобрать самостоятельно и добавить в ресурсы программы. Хранить информацию о выбранной иконке в виде атрибута заметки типа int (номер выбранной иконки)	Из метода Notebook::data() для роли Qt::DecorationRole необходимо вернуть соответствующий объект QIcon

2. Результаты выполнения заданий

2.1. Результат выполнения задания A0

1) Добавил в Qt Designer в меню «Help» пункт «Visit eCourses». Для реализации метода QDesktopServices::openUrl() подключил библиотеку QDesktopServices. Создал слот в форме и с помощью метода QDesktopServices::openUrl() открываю URL.
2) В метод MainWindow::displayAbout() добавил свои имя, фамилию, отчество, номер группы и зачётной книжки.
3) Добавил в Qt Designer в меню «File» пункт «Exit». В форме создал слот и использовал метод QCoreApplication::quit() для завершения программы, заблаговременно получил указатель на программу через метод QCoreApplication::instance().
4) Подключил библиотеку QMessageBox, в методе MainWindow::deleteNotes() создал диалоговое окно с помощью QMessageBox.
5) Добавил в Qt Designer в меню «File» пункт «Save As Text», создал его слот. Сделал проверку на наличие открытой записной книжки. Если ни какая записная книжка не открыта, то возвращаем сообщения об ошибке. Сделал проверку на пустоту названия файла. Если название файла пустое, то возвращаю результат false. Далее создал переменную QTextStream и передал в неё адрес переменной QFile. После чего сохранил записную книжку в выбранный файл, закрыл файл, установил выбранное имя файла в качестве текущего, сигнализировал о готовности и о сохранении записной книжки. После чего вернул результат true. В классе Notebook создал метод Notebook::SaveAsText(QTextStream &ost) const, в котором с помощью цикла записал номер заметки, общее количество заметок, заголовок и текст заметки.
6) Назначил сочетания клавиш в Qt Designer.
7) Добавил в Qt Designer пустое меню «Edit».
8) Добавил в Qt Designer в меню «Help» пункт «Lottery». Создал к нему слот и в классе MainWindow создал метод MainWindow::openLottery(). В нём через QMessageBox вывожу результаты лотереи, исходя из заранее сгенерированных значений. Если это победа, то приз берется из вектора призов.

2.2. Результат выполнения задания Б0

1) В метод EditNoteDialog::accept() добавил условие if(mUi->titleEdit->text().isEmpty() || mUi->plainTextEdit->toPlainText().isEmpty()), которое проверяет пустоту названия и содержания заметки. Если условие верно, то вывожу сообщение об ошибке с помощью QMessageBox, иначе вызываю метод QDialog::accept() для подтверждения диалога.
2) В классе Notebook создал метод Notebook::edit(QModelIndex index, QString s1, QString s2), в котором устанавливаю новый заголовок и содержание заметки. В классе EditNoteDialog создал метод EditNoteDialog::edit(QString s1, QString s2), с помощью которого устанавливаю новое содержание заметки в окне редактирования.

2.3. Результат выполнения задания В0

1) В методе MainWindow::refreshWindowTitle() использовал метод VisibleElements(), передавая в него значение isNotebookOpen(). С помощью подобной mUi->actionNew_Note->setEnabled(isNotebookOpen()) конструкции отключил все невостребованные элементы интерфейса.

2.4. Результат выполнения задания В1

1) В editnotedialog.ui добавил 3 радиокнопки, в ресурсы программы добавил 3 изображения формата .svg, на каждую радиокнопку поставил свое изображение. В класс EditNoteDialog добавил метод SetNumberRadiobutton(), с его помощью определяю нажатие на кнопку. В метод Notebook::*data*(const QModelIndex &index, int role) const добавил условие if (role == Qt::DecorationRole), с помощью которого устанавливается иконка заметки на лицевой панели. В методе Notebook::SaveAsText(QTextStream &ost) const добавил аргумент на сохранение иконки. В метод Notebook::edit(QModelIndex index, QString title, QString text, qint32 icon) добавлен новый параметр, также добавлена строчка для редактирования иконки. В класс Note добавил метод icon() const для установки выбранной иконки и метод setidIcon (qint32 number) для установки номера иконки.

3. Проверка результатов

3.1. Проверка результатов задания А0

1) Нажал на пункт «Visit eCourses», попал на сайт СФУ (Рисунок 1).

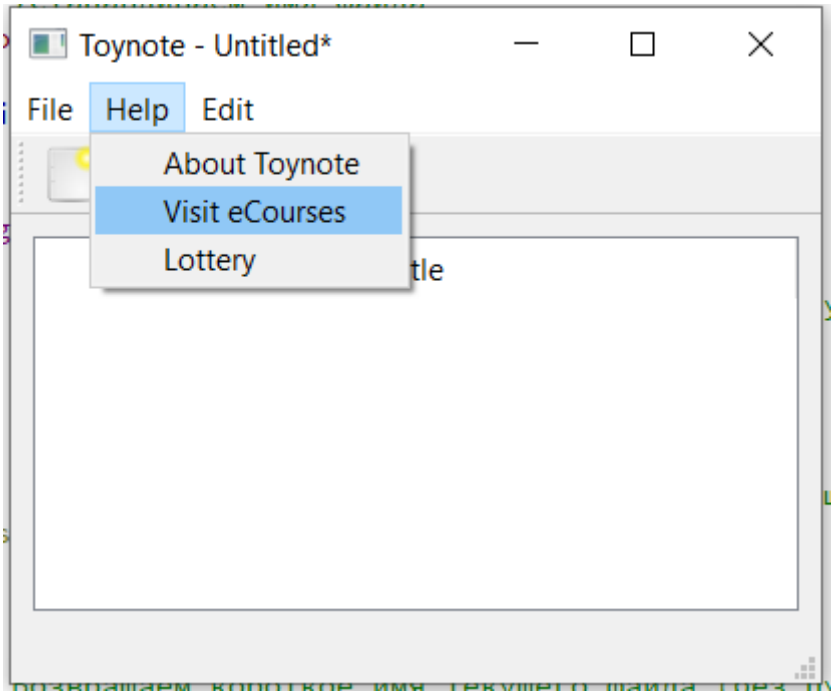

Рисунок 1 – Результат выполнения задания A0 1

2) Зашел в пункт «About Toynote» (Рисунок 2).

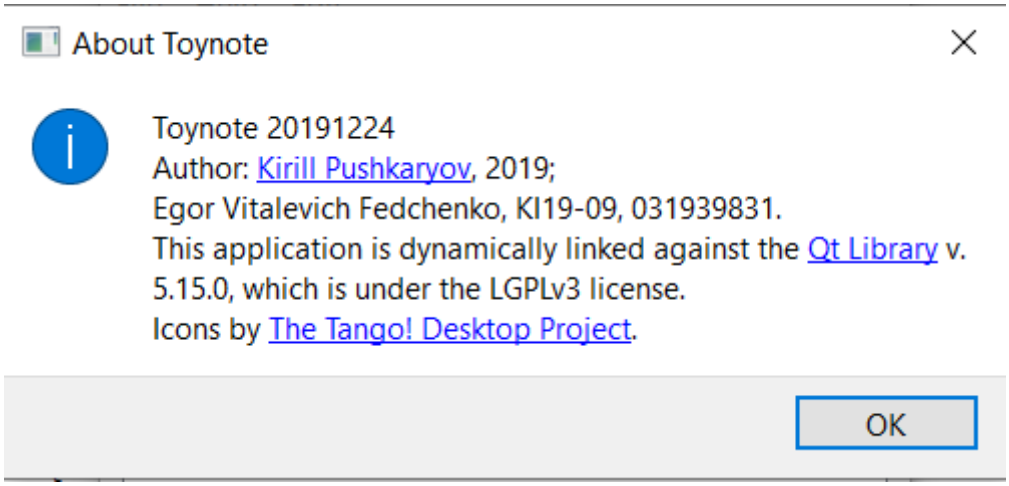

Рисунок 2 – Результат выполнения задания A0 2

3) Зашел во вкладку «File» и вышел из программы, используя пункт «Exit».
4) При удалении заметки из записной книжки появилось подтверждение удаления заметки (Рисунок 3).

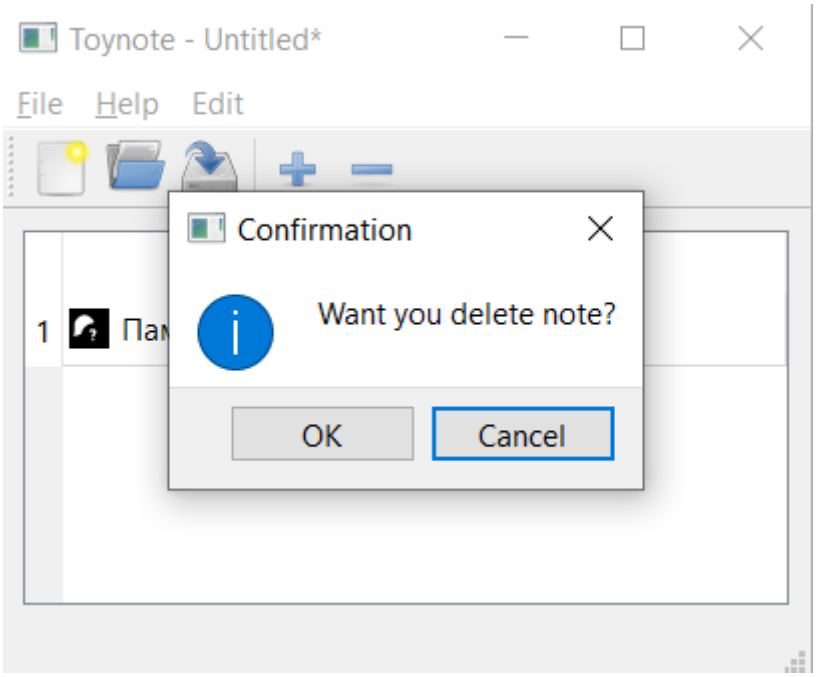

Рисунок 3 – Результат выполнения задания A0 4

5) При нажатии на пункт «Save As Text» заметка сохраняется в текстовом формате (Рисунок 4).

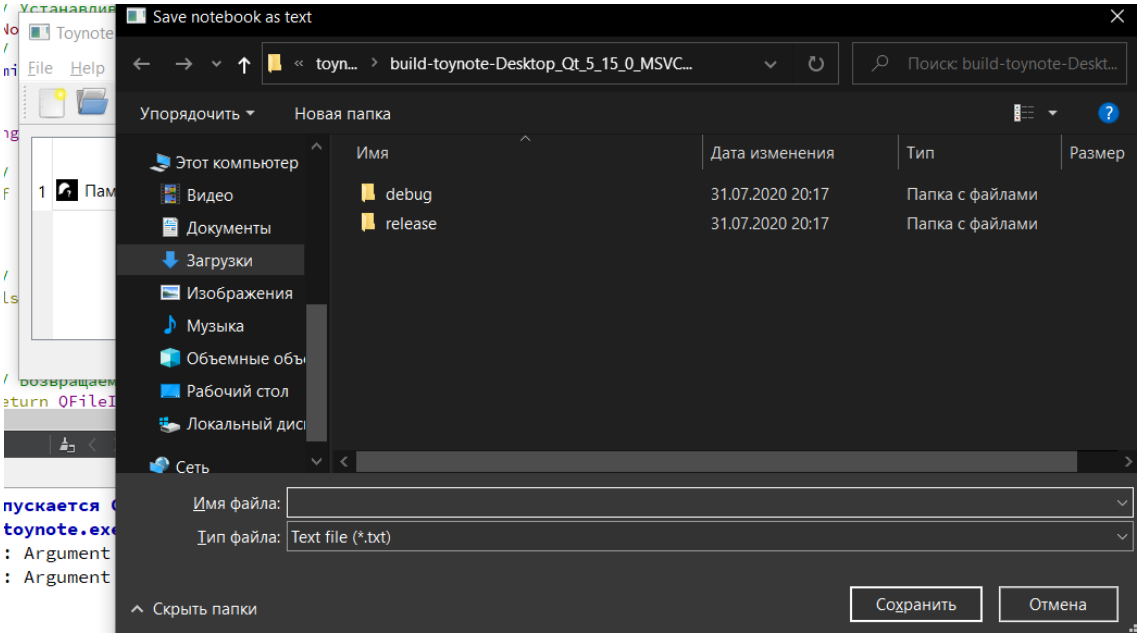

Рисунок 4 – Результат выполнения задания A0 5

6) Назначены сочетания клавиш для пунктов меню «File» (Рисунок 5).

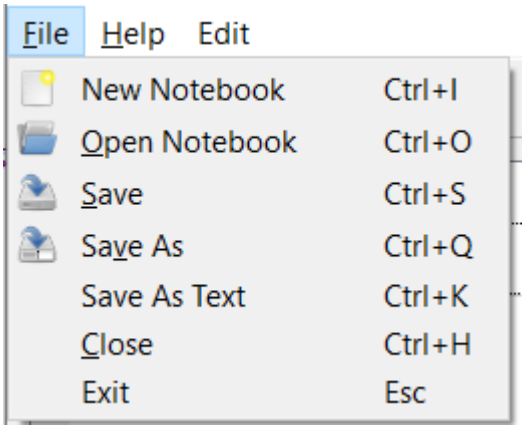

Рисунок 5 – Результат выполнения задания A0 6

7) Добавлено пустое меню «Edit».
8) Добавлен пункт «Lottety» (Рисунок 6).

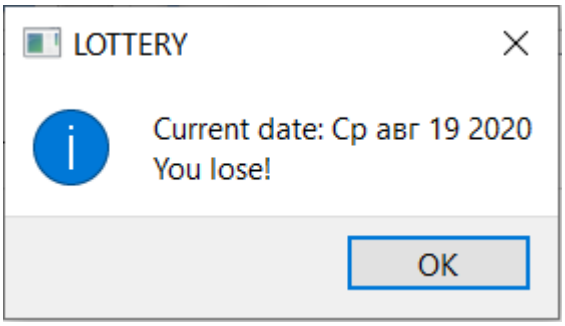

Рисунок 6 – Результат выполнения задания 8

3.2. Проверка результатов задания Б0

1) Запрещено создавать заметки с пустым заголовком или содержанием (Рисунок 7).

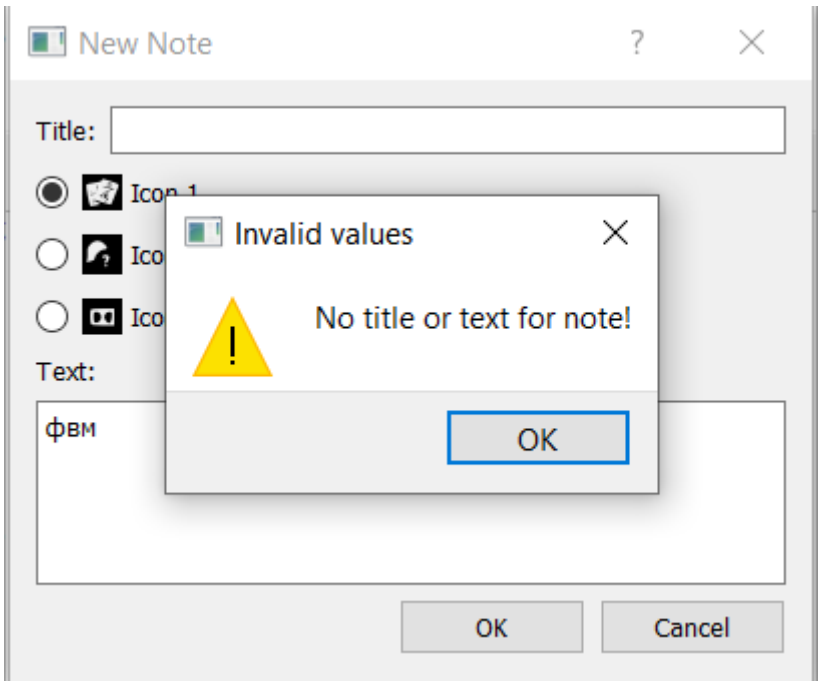

Рисунок 7 – Результат выполнения задания Б0 1

2) Добавлена возможность редактирования заметки (Рисунок 8).

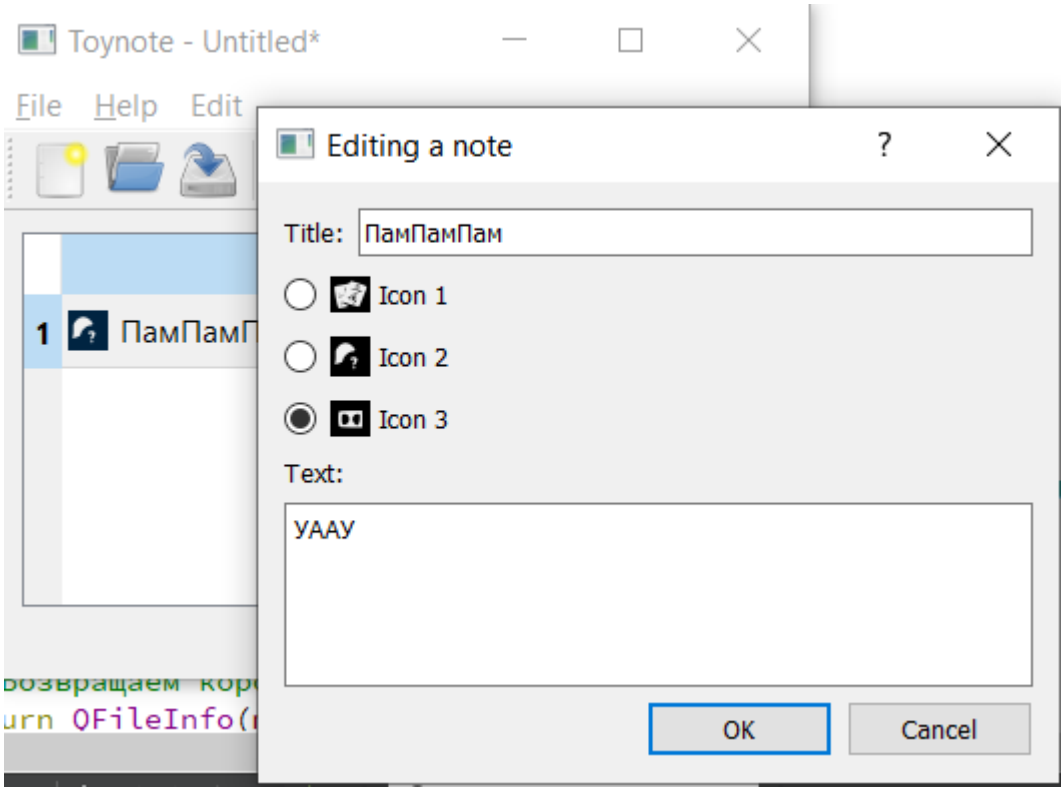

Рисунок 8 – Результат выполнения задания Б0 2

3.3. Проверка результатов задания B0

1) Добавлено отключение не имеющих смысла действий в главном окне программы.

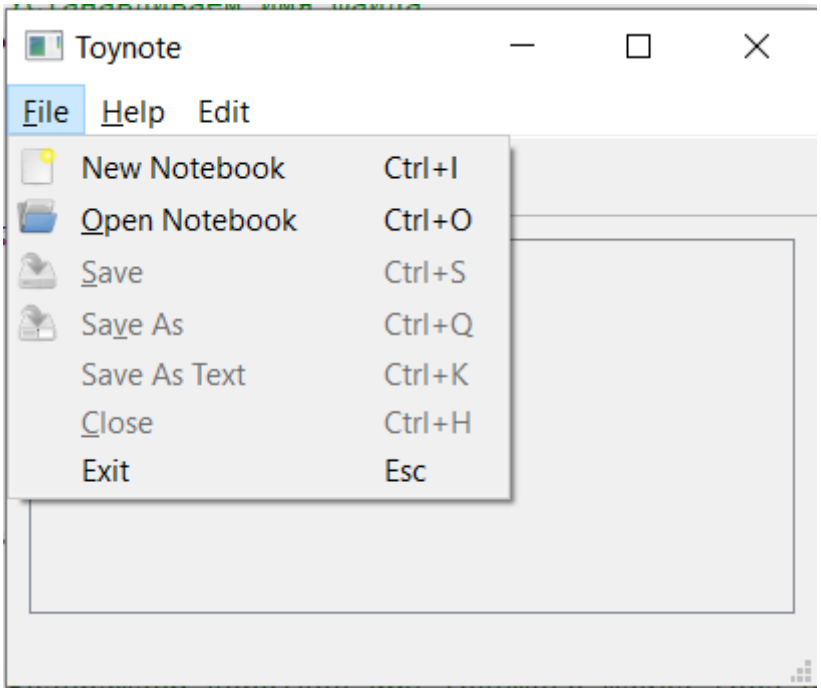

Рисунок 9 – Результат выполнения задания B0 1

3.4. Проверка результатов задания B1

1) Добавлена возможность установки иконки на заметку и ее редактирование (Рисунок 10 и Рисунок 11).

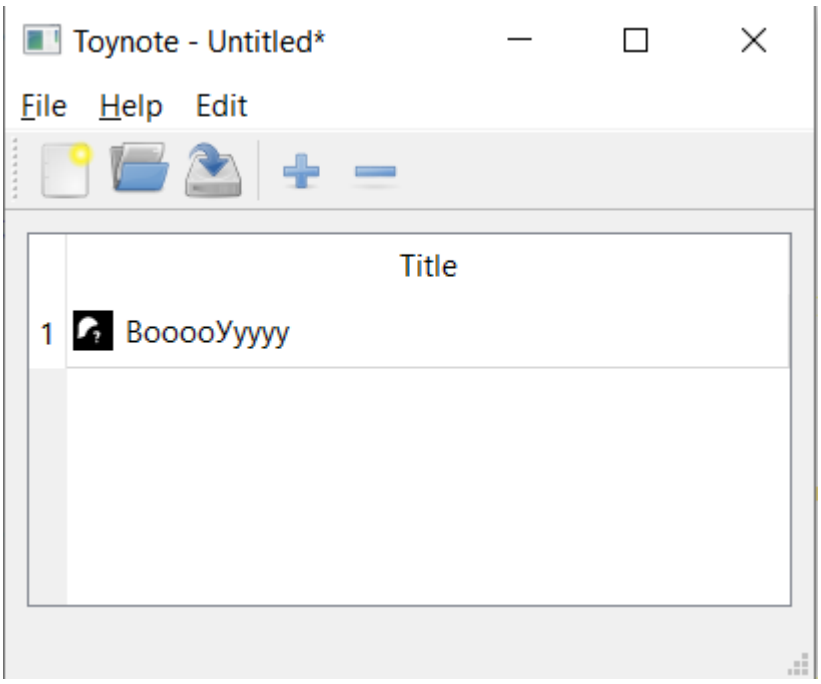

Рисунок 10 – Результат выполнения задания B1(1)

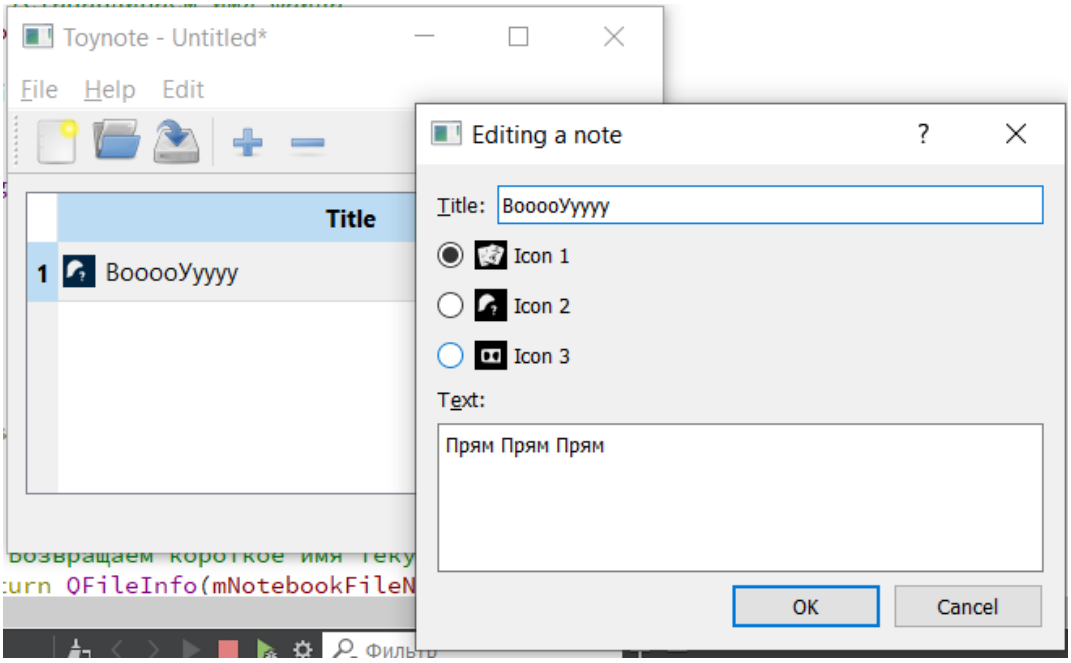

Рисунок – 11 Результат выполнения задания B1(2)

4. Список использованных источников

1) **E**-learning SibFU. – Режим доступа: https://e.sfu-kras.ru
2) Qt Documentation. – Режим доступа: https://doc.qt.io

ПРИЛОЖЕНИЕ А

Программная часть А0:

1.
```
void MainWindow::openURL()//mainwindows.cpp
{
 QDesktopServices::openUrl(QUrl("https://e.sfu-kras.ru"));
}
```

2.
```
"Egor Vitalevich Fedchenko, KI19-09, 031939831. <br>"//mainwindows.cpp
```

3.
```
void MainWindow::exit ()//mainwindows.cpp
{
  QCoreApplication::instance()->quit(); //получает указатель на программу и завершает ее
}
```

4.
```
QMessageBox delNod;//mainwindows.cpp
    delNod.setWindowTitle(tr("Confirmation").arg(Config::applicationName));
    delNod.setStandardButtons(QMessageBox::Ok | QMessageBox::Cancel);
    delNod.setDefaultButton(QMessageBox::Cancel);
    delNod.setIcon(QMessageBox::Information);
    delNod.setText(tr("Want you delete note?").arg(Config::applicationName));
    int ret = delNod.exec();
```

5.
```
bool MainWindow::saveAsText()//mainwindows.cpp
{
    // Если записная книжка не открыта, выдаём сообщение об этом
    if (!isNotebookOpen())
    {
        QMessageBox::warning(this, tr(Config::applicationName), tr("No open notebooks"));
        return false;
    }
    // Выводим диалог выбора файла для сохранения
    QString fileName = QFileDialog::getSaveFileName(this, tr("Save notebook as text"), QString(), tr("Text file (*.txt)"));
    // Если пользователь не выбрал файл, возвращаем false
    if (fileName.isEmpty())
    {
        return false;
    }
    QFile file(fileName);
```

```
    if(!file.open(QIODevice::WriteOnly))
    {
       return false;
    }
    QTextStream out(&file);
    // Сохраняем записную книжку в выбранный файл
    mNotebook->SaveAsText(out);
    file.close();
    // Устанавливаем выбранное имя файла в качестве текущего
    setNotebookFileName(fileName);
    // Сигнализируем о готовности
    emit notebookReady();
    // Сигнализируем о сохранении записной книжки
    emit notebookSaved();
    return true;
       }
```

6. Задание не предполагало написание кода.

7. Задание не предполагало написание кода.

8.
```
void MainWindow::openLottery() //mainwindows.cpp
{
   const qint32 numTickets = 20; // Участвующих билетов.
   const qint32 numWinTickets = 2; // Кол-во выигрышных билетов.

   QVector<QString> gifts = {tr("money"), tr("beer"), tr("new ticket"), tr("love"),
tr("tea")};

   qint32 randTicketNumber = QRandomGenerator::global()->generate()%numTickets;
   qint32 randGiftNumber = QRandomGenerator::global()->generate()%gifts.size();
   QMessageBox::information(this, tr("LOTTERY"), tr("Current date: %1\nYou %4")
  .arg(QDate::currentDate().toString())
  .arg(randTicketNumber>numWinTickets? tr("lose!"):tr("won!\nYour pot - a lot of %1!!")
  .arg(gifts[randGiftNumber])));
  }
```

Программная часть Б0:

1.
```
if (mUi->titleEdit->text().trimmed() == "" || mUi->plainTextEdit-
>toPlainText().trimmed() == "") //editnotedialog.cpp
        {
           QMessageBox::warning(this, tr("Invalid values"), tr("No title or text for note!"));
           return;
```

```
        }
```

2.
```
    void EditNoteDialog::edit(QString title, QString text) //editnotedialog.cpp
    {
        mUi->titleEdit->setText(title);
        mUi->plainTextEdit->setPlainText(text);
    }
```

3.
```
    void Notebook::edit(QModelIndex index, QString title, QString text, qint32 icon)
//notebook.cpp
    {
        mNotes[index.row()].setTitle(title);
        mNotes[index.row()].setText(text);
        mNotes[index.row()].setidIcon(icon);
        emit dataChanged(index, index);
    }
```

Программная часть B0:

1.
```
    void MainWindow::VisibleElements(bool visible)//mainwindows.cpp
    {
        mUi->actionSave->setEnabled(visible);
        mUi->actionDelete_Notes->setEnabled(visible);
        mUi->actionSave_As->setEnabled(visible);
        mUi->notesView->setEnabled(visible);
        mUi->actionNew_Note->setEnabled(visible);
        mUi->actionCloseNotebook->setEnabled(visible);
        mUi->actionSave_As_Text->setEnabled(visible);
    }
```

Программная часть B1:

1.
```
    if (role == Qt::DecorationRole)//notebook.cpp
        {
            switch (mNotes[index.row()].icon())
            {
            case 1:
                return QIcon(":/icons/Icon_1.svg");
                break;
            case 2:
                return QIcon(":/icons/Icon_2.svg");
                break;
            case 3:
                return QIcon(":/icons/Icon_3.svg");
                break;
```

```
            default:
                break;
        }
    }
```

2.
```
int EditNoteDialog::SetNumberRadiobutton()//editnotedialog.cpp
{
    QButtonGroup* buttongroup = new QButtonGroup;
    buttongroup->addButton(mUi->radioButton, 1);
    buttongroup->addButton(mUi->radioButton_2, 2);
    buttongroup->addButton(mUi->radioButton_3, 3);

    return buttongroup->checkedId();
}
```