Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №2
по дисциплине: **ООТПиСП**
Тема: Использование библиотеки элементов графического интерфейса Qt

Выполнил
студент 3 курса
Корнасевич И. Д.

Проверил
Булей Е. В.

Брест 2021

Цель работы Эта лабораторная работа проведёт вас через все этапы построения законченного приложения на Qt. Целью является понимание структуры приложения на Qt и получение опыта использования стандартной документации Qt.

Задание Написать текстовый редактор со следующим функционалом:

- Копирование, вставка, отмена и повторение действий
- Открытие файла, его сохранение
- Изменение шрифта
- Оповещение при выходе без сохранения.

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QCloseEvent>
#include <QMessageBox>
#include <QApplication>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    MainWindow(const QString &fileName=QString(), QWidget *parent=nullptr);
    ~MainWindow();

protected:
    void closeEvent(QCloseEvent *event);

private slots:
    void on_actionNew_triggered();

    void on_actionSelectFont_triggered();

    void on_actionAbout_triggered();

    void on_actionOpen_triggered();

    bool saveFile();

    bool saveFileAs();

private:
    QString fileName;
    Ui::MainWindow *ui;

    void loadFile(const QString &fileName);

    void setFileName(const QString &fileName);
};
#endif // MAINWINDOW_H
```

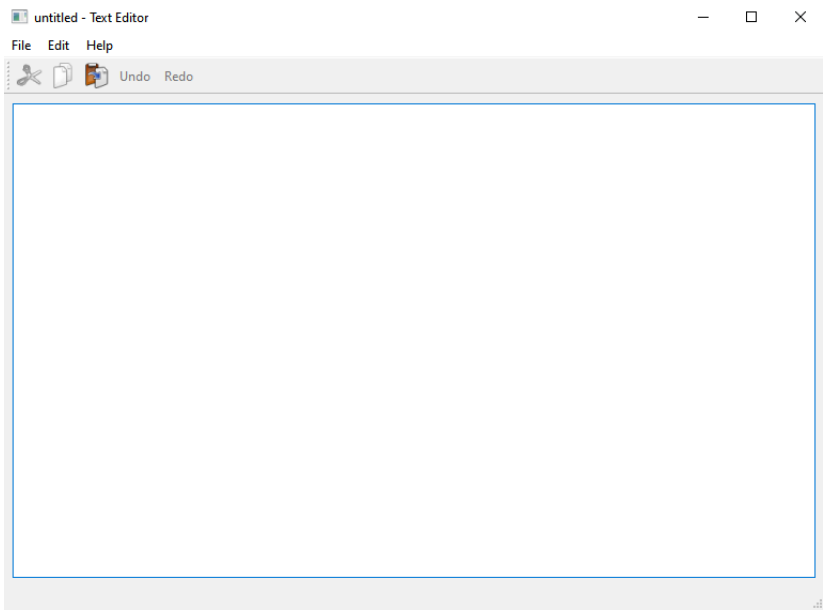
mainwindow.cpp

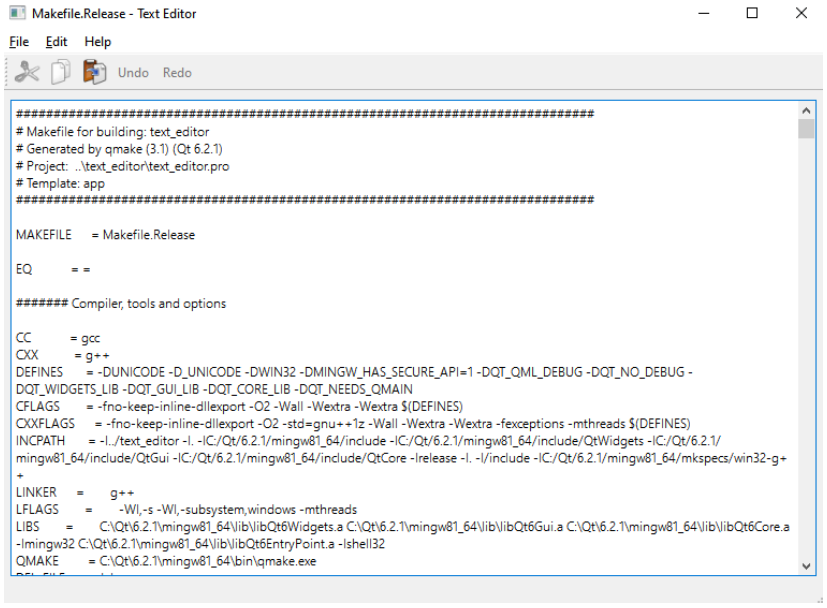
```
#include "mainwindow.h"
#include "ui_mainwindow.h"

#include <QFile>
#include <QFontDialog>
#include <QSettings>
#include <QTextEdit>
#include <Qt>
```

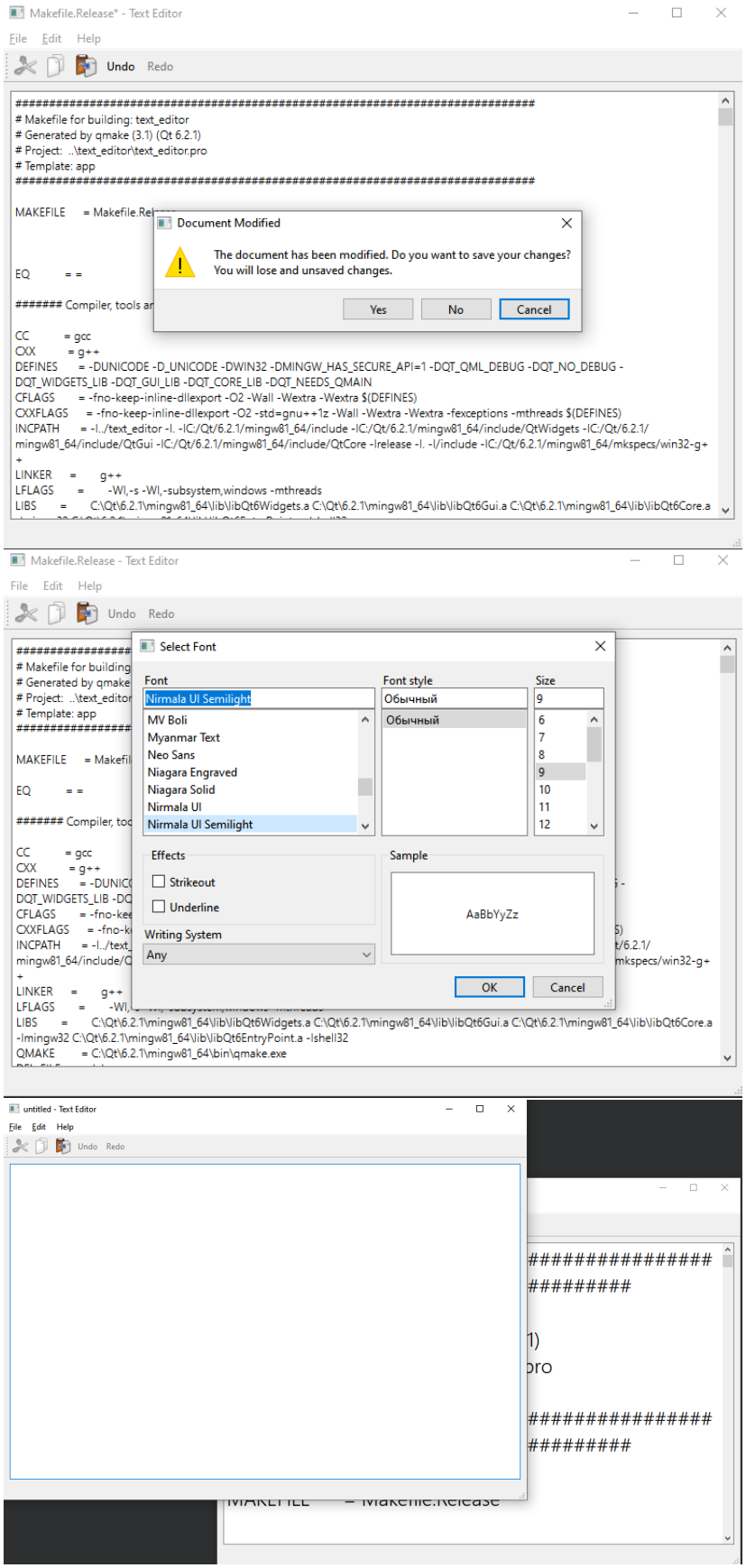
```cpp
#include <QFileDialog>

MainWindow::MainWindow(const QString &fileName, QWidget *parent): QMainWindow(parent), ui(new Ui::
    MainWindow)
{
    ui->setupUi(this);

    loadFile(fileName);

    QSettings settings;

    auto font = settings.value("viewFont", QApplication::font()).value<QFont>();
    ui->textEdit->setFont(font);

    connect(ui->actionClose, &QAction::triggered, this, &QMainWindow::close);
    connect(ui->actionExit, &QAction::triggered, &QApplication::closeAllWindows);
    connect(ui->textEdit, &QTextEdit::textChanged, [this](){this->setWindowModified(true);});

    connect(ui->actionAboutQT, &QAction::triggered, &QApplication::aboutQt);

    connect(ui->actionCut, &QAction::triggered, ui->textEdit, &QTextEdit::cut);
    connect(ui->actionCopy, &QAction::triggered, ui->textEdit, &QTextEdit::copy);
    connect(ui->actionPaste, &QAction::triggered, ui->textEdit, &QTextEdit::paste);
    connect(ui->actionUndo, &QAction::triggered, ui->textEdit, &QTextEdit::undo);
    connect(ui->actionRedo, &QAction::triggered, ui->textEdit, &QTextEdit::redo);

    ui->actionCut->setEnabled(false);
    ui->actionCopy->setEnabled(false);
    ui->actionUndo->setEnabled(false);
    ui->actionRedo->setEnabled(false);

    connect(ui->textEdit, &QTextEdit::copyAvailable, ui->actionCut, &QAction::setEnabled);
    connect(ui->textEdit, &QTextEdit::copyAvailable, ui->actionCopy, &QAction::setEnabled);
    connect(ui->textEdit, &QTextEdit::undoAvailable, ui->actionUndo, &QAction::setEnabled);
    connect(ui->textEdit, &QTextEdit::redoAvailable, ui->actionRedo, &QAction::setEnabled);

    connect(ui->actionSaveAs, &QAction::triggered, this, &MainWindow::saveFileAs);
    connect(ui->actionSave, &QAction::triggered, this, &MainWindow::saveFile);
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::closeEvent(QCloseEvent *event)
{
    if (this->isWindowModified()){
        auto button = QMessageBox::warning(this, "Document Modified",
                                           "The document has been modified. "
                                           "Do you want to save your changes?\n"
                                           "You will lose and unsaved changes.",
                                           QMessageBox::Yes | QMessageBox::No | QMessageBox::Cancel,
                                           QMessageBox::Cancel);
        switch(button){
            case QMessageBox::Yes:
                if (saveFile()){
                    event->accept();
                } else {
                    event->ignore();
                }
                break;
            case QMessageBox::No:
                event->accept();
                break;
            case QMessageBox::Cancel:
                event->ignore();
                break;
            default:
                event->ignore();
        }

    } else {
        event->accept();
    }
}


void MainWindow::on_actionNew_triggered()
{
    auto editor = new MainWindow;
    editor->show();
```

```
90  }
91
92
93  void MainWindow::on_actionSelectFont_triggered()
94  {
95      bool isOk;
96      auto font = QFontDialog::getFont(&isOk, ui->textEdit->font(), this);
97      if (isOk){
98          QSettings settings;
99          settings.setValue("viewFont", font);
100         ui->textEdit->setFont(font);
101     }
102 }
103
104
105 void MainWindow::on_actionAbout_triggered()
106 {
107     QMessageBox::about(this, "about", "about");
108 }
109
110 void MainWindow::loadFile(const QString &fileName){
111     if (fileName.isEmpty()){
112         setFileName(QString());
113         return;
114     }
115
116     QFile file(fileName);
117
118     auto isOpen = file.open(QIODevice::ReadOnly | QIODevice::Text);
119     if (!isOpen){
120         QMessageBox::warning(this, "File error", "The file is not opened");
121         setFileName(QString());
122         return;
123     }
124
125     ui->textEdit->setText(file.readAll());
126
127     file.close();
128     setFileName(fileName);
129     setWindowModified(false);
130 }
131
132 void MainWindow::setFileName(const QString &fileName)
133 {
134     auto title = QString("%1[*] - %2")
135             .arg(fileName.isEmpty()?"untitled":QFileInfo(fileName).fileName(), QApplication::
    applicationName());
136     this->setWindowTitle(title);
137     this->fileName = fileName;
138 }
139
140
141 void MainWindow::on_actionOpen_triggered()
142 {
143     QString fileName = QFileDialog::getOpenFileName(this,
144         "Open document", QDir::currentPath());
145     if (!fileName.isEmpty() && !isWindowModified()){
146         loadFile(fileName);
147     } else {
148         auto editor = new MainWindow(fileName);
149         editor->show();
150     }
151 }
152
153 bool MainWindow::saveFile()
154 {
155     if (fileName.isEmpty()){
156         return saveFileAs();
157     }
158     QFile file(fileName);
159     auto isOpen = file.open(QIODevice::WriteOnly | QIODevice::Text);
160     if (!isOpen){
161         QMessageBox::warning(this, "File error", "The file is not opened");
162         setFileName(QString());
163         return false;
164     }
165
166     file.write(ui->textEdit->toPlainText().toUtf8());
167
168     file.close();
169     setWindowModified(false);
170     return true;
```

```
171 }
172
173 bool MainWindow::saveFileAs()
174 {
175     QString fileName = QFileDialog::getSaveFileName(this, "Save document",
176        this->fileName.isEmpty()?QDir::currentPath():this->fileName);
177     if (fileName.isEmpty()){
178         return false;
179     }
180     setFileName(fileName);
181     return saveFile();
182 }
```







Вывод Разработан текстовый редактор с возможностью сохранять и загружать файлы, изменять шрифт, копировать текст, вставлять его, отменять и повторять свои действия.