

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Кафедра «Компьютерная безопасность»

**ОТЧЕТ
К ЛАБОРАТОРНОЙ РАБОТЕ №5**

по дисциплине

«Языки программирования»

Работу выполнил
студент группы СКБ-201

подпись, дата

Г.П. Кашкин

Работу проверил

подпись, дата

М.Ю. Моница

Содержание

1	Постановка задачи	3
2	Алгоритм решения задачи	6
2.1	Текстовый редактор	6
2.2	Подсветка синтаксиса	6
2.3	Поиск и замена	6
2.4	Окно приложения	6
3	Выполнение задания	7
3.1	Задание 1	7
3.2	Текстовый редактор	7
3.3	Подсветка синтаксиса	7
3.4	Поиск и замена	7
3.5	Окно приложения	7
4	Получение исполняемых модулей	7
	Приложение А	8
	Приложение Б	9
	Приложение В	10
	Приложение Г	19
	Приложение Д	31

1 Постановка задачи

Тема: Создание приложений. Меню, диалоги, работа с файлами.

Задание на 6 баллов.

Разработать графическое приложение с использованием библиотеки Qt – текстовый редактор:

- а) с подсветкой текущей строки;
- б) с нумерацией строк;
- в) с подсветкой синтаксиса (переключение): Си (стандарт 2011), Си++ (стандарт 2014)

Заголовок окна должен содержать имя редактируемого файла (ограниченное 32 символами + троеточие) и признак того что файл был изменен (звездочка в начале имени) с момента последнего сохранения.

Окно (наследуется от QMainWindow) и содержит главное меню состоящее из пунктов:

а) Файл [кнопки]

- 1) Новый
- 2) Открыть
- 3) Сохранить
- 4) Сохранить как
- 5) Выход

б) Правка [кнопки]

- 1) Отменить
- 2) Повторить
- 3) Копировать
- 4) Вырезать
- 5) Вставить
- 6) Найти
- 7) Найти и заменить
- 8) Выделить все

в) Формат

- 1) Перенос по словам [галочка]
- 2) Выбор шрифта – открывается модальный диалог выбора шрифта

Вид [галочки, где не указано иное]

- 1) Выбор цвета фона [кнопка] – открывает модальное окно выбора цвета
- 2) Выбор цвета текущей строки [кнопка] – открывает модальное окно выбора цвета
- 3) Вкл/Выкл отображения нумерации строк
- 4) Вкл/Выкл отображения панели инструментов

- 5) Вкл/Выкл отображения строки состояния
- 6) Вкл/Выкл подсветки синтаксиса
- 7) Выбор синтаксиса (для подсветки) [дочернее меню] – доступно всегда, один синтаксис в дочернем меню выбран всегда
- 8) Выбор/Редактирование стиля подсветки [дочернее меню] – для текущего синтаксиса, по умолчанию выбран Default
 - i. Изменить [кнопка] – измененный стиль сохраняется в файл, имя файла становится именем стиля, стиль становится активным
 - ii. Загрузка стиля из файла [кнопка] – имя файла становится именем стиля, стиль становится активным
 - iii. Обязательная кнопка Default
 - iv. доступные стили [кнопки – перечисляются все стили, которые были обнаружены]
- г) Справка
 - 1) О программе – открывает модальное окно, содержащее фото и имя автора, дату сборки, версию Qt с которой собиралось, версию Qt с которой запущено, кнопку закрывающую окно

Ниже главного меню располагается панель инструментов (отображение которой контролируется в меню Вид) с кнопками (с картинками, текстовое описание во всплывающей подсказке):

- а) Новый документ
- б) Открыть
- в) Сохранить
- г) Отменить
- д) Повторить
- е) Копировать
- ж) Вырезать
- з) Вставить
- и) Найти / Найти и заменить (как выпадающая кнопка) – открывающая (немодальное) диалоговое окно

В центральной части окна располагается область для редактирования текста. При нажатии левой кнопки курсор вставляется в позицию. При двойном нажатии левой кнопки выделяется слово под курсором. При нажатии правой кнопки (далее – если нет=*, есть=** выделения) курсор вставляется в позицию и выдается контекстное меню (кнопки могут быть неактивны): отменить, повторить, выделить*, выделить строку*, копировать**, вырезать**, вставить (** или если есть текст в буфере обмена), удалить**, выделить все.

Нижнюю часть окна занимает строка состояния. Информация разделена на три столбца: текущая позиция курсора (строка:столбец); время (и дата если другие сутки) последней операции (сохранения/изменения); количество строк, слов, символов, размер в килобайтах.

Задание на +1 балл. Реализовать подсветку синтаксиса Си (стандарт 2018), Си++17, Си++20

Задание на +1 балл. Реализовать сохранение настроек приложения в ini-файл.

Задание на +2 балла. Реализовать сохранение истории изменения редактируемого файла (бесконечную очередь команд повторения/отмены) во внешнем файле (обязательно хранить дату и время изменения; при отсутствии файла – создавать, при наличии – дописывать).

2 Алгоритм решения задачи

2.1 Текстовый редактор

На базе QPlainTextEdit был создан кастомный класс CodeEditor с базовой логикой текстового редактора и полями под указатели на класс подсветки синтаксиса и классы поиска/замены.

2.2 Подсветка синтаксиса

Реализован класс подсветки синтаксиса как наследник QSyntaxHighlighter включающий в себя интерфейс для конфигурации, поля типа подсветки (через enum class с акомпонирующим массивом строк-названий для каждого значения енама). Правила подсветки хранятся в виде строк-литералов и передаются в QStringList паттернов.

2.3 Поиск и замена

Модуль позволяет делать гибкий поиск и замену строк в тексте через всплывающее модальное окно.

2.4 Окно приложения

Окно приложения реализовано как отдельный класс, создает требуемый интерфейс и соединяет необходимые пары слотов и сигналов.

3 Выполнение задания

3.1 Задание 1

3.2 Текстовый редактор

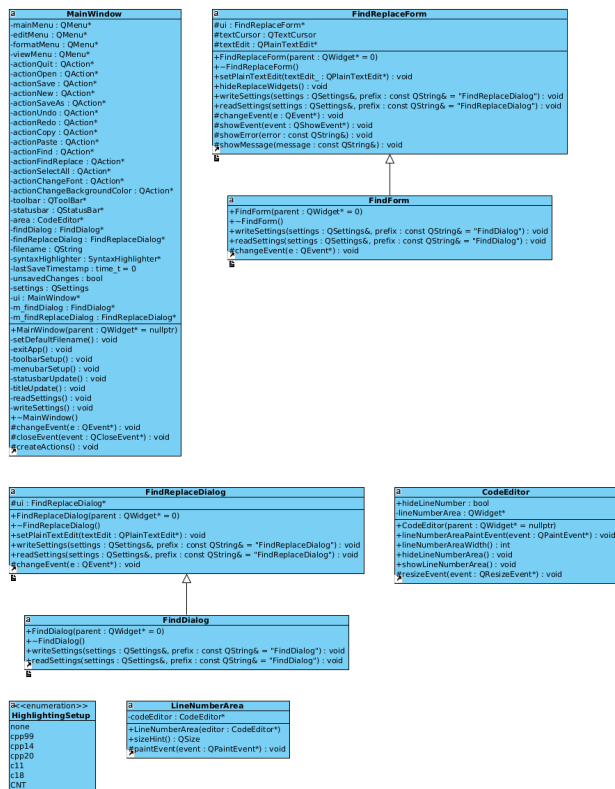


Рис. 1. UML 2.0 diagram for gCodeEditor

Класс текстового редактора был пронаследован от QPlainTextEdit. Из добавочной логики можно отметить отключаемую нумерацию строк, которая реализована как отдельная зона левее зоны текстового редактора, ширина этой зоны вычисляется динамически в зависимости от количества строк.

3.3 Подсветка синтаксиса

Класс подсветки синтаксиса был пронаследован от `QSyntaxHighlighter`, был добавлен интерфейс конфигурации, позволяющий изменять стиль подсветки и выбирать конкретную версию языка.

3.4 Поиск и замена

Для решения данной задачи был найден модуль диалога поиска замены, который был в свою очередь доработан для совместимости с QPlainTextEdit (<https://github.com/Lord-KA/QtFindReplaceDialogGitHub>). Модуль подтягивается прямо с github по тегу при помощи CMake и собирается в статическую библиотеку, которая в свою очередь подключается к основному проекту.

3.5 Окно приложения

Окно приложения реализовано как отдельный класс, создает требуемый интерфейс, создает и конфигурирует вспомогательные объекты такие как SyntaxHighlighter, CodeEditor и FindReplaceDialog.

4 Получение исполняемых модулей

Для всего проекта использовалась система сборки **cmake**. В конфигурации системы сборки прописаны пять режимов компиляции: `basic`, `sanitizer`, `release`, `debug` и `coverage` с разными наборами флагов компиляции (использование посредством флага `-DCMAKE_BUILD_TYPE=*`), в каждом из них прописано использование требуемого стандарта `c++20`, а для проверки на покрытие тестов специфицируется компилятор `clang++`, из-за отсутствия универсального набора флагов у основных компиляторов. Помимо этого **cmake** автоматически скачивает с `github` и подключает библиотеку с реализацией `FindReplaceDialog`.

Приложение А

subsectionФайл CMakeLists.txt

```
1 cmake_minimum_required(VERSION 3.1.0)
2
3 project(Lab_5)
4
5 set(CMAKE_AUTOMOC ON)
6 set(CMAKE_AUTORCC ON)
7 set(CMAKE_AUTOUIC ON)
8
9 if(CMAKE_VERSION VERSION_LESS "3.7.0")
10     set(CMAKE_INCLUDE_CURRENT_DIR ON)
11 endif()
12
13 set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=c++20 -g -D EXTRA_VERBOSE"
    CACHE STRING "Comment" FORCE)
14 set(CMAKE_CXX_FLAGS_SANITIZE "${CMAKE_CXX_FLAGS} -Wpedantic -Wall -Wextra
    -Wformat=2 -fsanitize=address,undefined -g" CACHE STRING "Comment" FORCE
    )
15
16 find_package(Qt5 COMPONENTS Widgets REQUIRED)
17
18 include(FetchContent)
19
20 FetchContent_Declare(
21     frdialog
22     GIT_REPOSITORY https://github.com/lord-ka/QtFindReplaceDialog.git
23 )
24
25 FetchContent_GetProperties(frdialog)
26 if(NOT frdialog_POPULATED)
27     FetchContent_Populate(frdialog)
28     add_subdirectory(${frdialog_SOURCE_DIR}/dialogs)
29 endif()
30
31
32 add_executable( run
33     main.cpp
34     MainWindow.cpp
35     TextEditor.cpp
36     MainWindow.hpp
37     TextEditor.hpp
38     SyntaxHighlighter.hpp
39     SyntaxHighlighter.cpp
40 )
41
42 target_link_libraries(run Qt5::Widgets)
43 target_link_libraries(run QtFindReplaceDialog)
```


Приложение Б

subsectionФайл main.cpp

```
1 #include "MainWindow.hpp"
2 #include <QtWidgets/QApplication>
3
4 int main(int argc, char *argv[])
5 {
6     QApplication app(argc, argv);
7     MainWindow window;
8     window.show();
9     return app.exec();
10 }
```

Приложение В

В.1 Файл MainWindow.hpp

```
1 #pragma once
2
3 #include <QVector>
4 #include <QPushButton>
5 #include <QMenuBar>
6 #include <QCoreApplication>
7 #include <QContextMenuEvent>
8 #include <QToolBar>
9 #include <QPainter>
10 #include <QMainWindow>
11 #include <QString>
12 #include <QFileDialog>
13 #include <QFile>
14 #include <QMessageBox>
15 #include <QHBoxLayout>
16 #include <QComboBox>
17 #include <QFont>
18 #include <QSpinBox>
19 #include <QCheckBox>
20 #include <QWidgetAction>
21 #include <QColorDialog>
22 #include <QStatusBar>
23
24 #include <iostream>
25 #include <string>
26 #include <fstream>
27 #include <chrono>
28 #include <ctime>
29
30 #include "TextEditor.hpp"
31 #include "SyntaxHighlighter.hpp"
32
33 #include "finddialog.h"
34 #include "findreplacedialog.h"
35
36 static const char DEFAULT_FILENAME[] = "untitled";
37 static const char DEFAULT_FILETYPE[] = "txt";
38
39 static const QFont::StyleHint DEFAULT_FONT = QFont::Times;
40
41 class MainWindow : public QMainWindow
42 {
43     Q_OBJECT
44
45 public:
46     MainWindow(QWidget *parent = nullptr);
47
48 private:
49     void setDefaultFilename();
```

```

50     void exitApp();
51
52     void toolbarSetup();
53     void menubarSetup();
54
55     void statusBarUpdate();
56     void titleUpdate();
57
58 public slots:
59     void quit();
60     void openFile();
61     void newFile();
62     void saveFile();
63     void saveAsFile();
64
65     void changeFont();
66
67     void changeBackgroundColor();
68
69 private:
70     QMenu *mainMenu, *editMenu, *formatMenu, *viewMenu;
71     QAction *actionQuit, *actionOpen, *actionSave, *actionNew, *
        actionSaveAs;
72     QAction *actionUndo, *actionRedo, *actionCopy, *actionPaste, *
        actionFind, *actionFindReplace, *actionSelectAll;
73     QAction *actionChangeFont;
74     QAction *actionChangeBackgroundColor;
75
76     QToolBar *toolbar;
77     QStatusBar *statusbar;
78     CodeEditor *area;
79     FindDialog *findDialog;
80     FindReplaceDialog *findReplaceDialog;
81     QString filename;
82
83     SyntaxHighlighter *syntaxHighlighter;
84
85     std::time_t lastSaveTimestamp = 0;
86
87     bool unsavedChanges;
88 };

```

В.2 Файл MainWindow.cpp

```

1 #include "MainWindow.hpp"
2
3 MainWindow::MainWindow(QWidget* parent)
4     : QMainWindow(parent)
5 {
6     toolbarSetup();
7
8     menubarSetup();
9
10    area = new CodeEditor();

```

```

11
12     statusbar = new QStatusBar(this);
13     setStatusBar(statusbar);
14
15     findDialog = new FindDialog(this);
16     findDialog->setModal(false);
17     findDialog->setPlainTextEdit(area);
18
19     findReplaceDialog = new FindReplaceDialog(this);
20     findReplaceDialog->setModal(false);
21     findReplaceDialog->setPlainTextEdit(area);
22
23     setCentralWidget(area);
24     setMinimumSize(1280, 720);
25
26     setDefaultFilename();
27
28     syntaxHighlighter = new SyntaxHighlighter(area->document(),
HighlightingSetup::none, "tomorrow");
29
30     unsavedChanges = false;
31     titleUpdate();
32     statusBarUpdate();
33     connect(area, &QPlainTextEdit::textChanged, this, [this]() {
statusBarUpdate(); unsavedChanges = true; titleUpdate();});
34 }
35
36 void MainWindow::toolbarSetup()
37 {
38     toolbar = addToolBar("ToolBar");
39     toolbar->addAction("New", this, SLOT(newFile()));
40     toolbar->addAction("Open", this, SLOT(openFile()));
41     toolbar->addAction("Undo", this, [& this]() {area->undo();});
42     toolbar->addAction("Redo", this, [& this]() {area->redo();});
43     toolbar->addAction("Copy", this, [& this]() {area->copy();});
44     toolbar->addAction("Paste", this, [& this]() {area->paste();});
45     toolbar->addAction("Find", this, [& this]() {findDialog->
show();});
46     toolbar->addAction("Find and Replace", this, [& this]() {
findReplaceDialog->show();});
47 }
48
49 void MainWindow::menubarSetup()
50 {
51     /* Main Menu setup */
52     mainMenu = menuBar()->addMenu("File");
53     actionQuit = mainMenu->addAction("Quit", this, SLOT(quit()));
54     actionOpen = mainMenu->addAction("Open", this, SLOT(openFile()));
55     actionNew = mainMenu->addAction("New", this, SLOT(newFile()));
56     actionSave = mainMenu->addAction("Save", this, SLOT(saveFile()));
57     actionSaveAs = mainMenu->addAction("Save as", this, SLOT(saveAsFile()));
58
59     /* Edit Menu setup */

```

```

60     editMenu = menuBar()->addMenu("Edit");
61     actionUndo = editMenu->addAction("Undo", this, [&, this](){area->undo
();});
62     actionRedo = editMenu->addAction("Redo", this, [&, this](){area->redo
();});
63     actionCopy = editMenu->addAction("Copy", this, [&, this](){area->copy
();});
64     actionPaste = editMenu->addAction("Paste", this, [&, this](){area->
paste();});
65     actionFind = editMenu->addAction("Find", this, [&,
this](){findDialog->show();});
66     actionFindReplace = editMenu->addAction("Find and Replace", this, [&,
this](){findReplaceDialog->show();});
67     actionSelectAll = editMenu->addAction("Select All", this, [&,
this](){area->selectAll();});
68
69     /* Format Menu setup */
70     formatMenu = menuBar()->addMenu("Format");
71     actionChangeFont = formatMenu->addAction("Change font", this, SLOT(
changeFont()));
72     QWidgetAction *actionWrapperBox = new QWidgetAction(formatMenu);
73     QCheckBox *wrapperBox = new QCheckBox(formatMenu);
74     wrapperBox->setText("Enable wrapping");
75     actionWrapperBox->setDefaultWidget(wrapperBox);
76     connect(wrapperBox, &QCheckBox::stateChanged, this, [this, wrapperBox
]()
77     {
78         area->setLineWrapMode((QPlainTextEdit::LineWrapMode)!
wrapperBox->checkState());
79     });
80     formatMenu->addAction(actionWrapperBox);
81
82     /* View Menu setup */
83     viewMenu = menuBar()->addMenu("View");
84     actionChangeBackgroundColor = viewMenu->addAction("Change background
color", this, SLOT(changeBackgroundColor()));
85     QWidgetAction *actionHideLineNumBox = new QWidgetAction(formatMenu);
86     QCheckBox *hideLineNumBox = new QCheckBox(formatMenu);
87     hideLineNumBox->setText("Hide line numbers");
88     actionHideLineNumBox->setDefaultWidget(hideLineNumBox);
89     connect(hideLineNumBox, &QCheckBox::stateChanged, this, [this,
hideLineNumBox]()
90     {
91         if (hideLineNumBox->checkState())
92             area->hideLineNumberArea();
93         else
94             area->showLineNumberArea();
95         area->updateLineNumberAreaWidth();
96     });
97     viewMenu->addAction(actionHideLineNumBox);
98     viewMenu->addAction(toolbar->toggleViewAction());
99     /* Highlighting setup */ //TODO
100    QActionGroup *highlightingGroup = new QActionGroup(viewMenu);

```

```

101     for (int i = 0; i < static_cast<int>(HighlightingSetup::CNT); ++i) {
102         QAction *action = new QAction(HighlightingSetups[i],
highlightingGroup);
103         connect(action, &QAction::triggered, this, [this, i]() { this->
syntaxHighlighter->setSyntax(static_cast<HighlightingSetup>(i)); });
104         action = highlightingGroup->addAction(viewMenu->addAction(
HighlightingSetups[i], this, [this, i]() { syntaxHighlighter->setSyntax(
static_cast<HighlightingSetup>(i)); }));
105         highlightingGroup->addAction(action);
106         if (i == 0)
107             action->setChecked(true);
108     }
109     highlightingGroup->setVisible(true);
110 }
111
112 void MainWindow::statusbarUpdate()
113 {
114     QTextCursor cursor = area->textCursor();
115     size_t line = cursor.blockNumber() + 1;
116     size_t pos = cursor.positionInBlock() + 1;
117     cursor.movePosition(QTextCursor::End);
118     size_t lineCount = cursor.blockNumber() + 1;
119     size_t charsCount = cursor.position() + 1;
120     size_t wordCount = area->toPlainText().split(QRegExp("(\\s|\\n|\\r)+"),
121                                                     QString::SkipEmptyParts).
count();
122
123
124     #ifdef EXTRA_VERBOSE
125         fprintf(stderr, "line = %lu, pos = %lu\\n", line, pos);
126     #endif
127
128     std::tm *now = std::localtime(&lastSaveTimestamp);
129     QString message = QString::number(line) + ":" +
QString::number(pos) +
130         " | " + QString::number(now->tm_hour) + ":" +
QString::number(now->tm_min) + ":" + QString::number(now->tm_sec) +
131         " | l:" + QString::number(lineCount) + " | w:" +
QString::number(wordCount) + " | c:" + QString::number(charsCount) +
132         " | Kb:" + QString::number(charsCount / 1024);
133     statusBar->showMessage(message);
134 }
135
136 void MainWindow::titleUpdate()
137 {
138     QString title = filename;
139     if (unsavedChanges)
140         title += "*";
141     setWindowTitle(title);
142 }
143
144 void MainWindow::changeBackgroundColor()

```

```

145 {
146     QColorDialog *dialog = new QColorDialog(QColor("white"), this);
147     dialog->exec();
148
149     QPalette p = area->palette();
150     p.setColor(QPalette::Active, QPalette::Base, dialog->selectedColor());
151     area->setPalette(p);
152     area->setBackgroundVisible(false);
153 }
154
155 void MainWindow::changeFont()
156 {
157     #ifdef EXTRA_VERBOSE
158         std::cerr << "Changing font to ";
159     #endif
160     QFont currentFont = area->document()->defaultFont();
161     int fontSize = currentFont.pointSize();
162
163     QDialog *askFont = new QDialog(this);
164     askFont->setLayout(new QHBoxLayout());
165
166     QComboBox *fontBox = new QComboBox(askFont);
167     fontBox->addItem("Helvetica");
168     fontBox->addItem("Times");
169     fontBox->addItem("Courier");
170     fontBox->addItem("OldEnglish");
171     fontBox->addItem("System");
172     fontBox->addItem("Any");
173
174     fontBox->setCurrentIndex((int)currentFont.styleHint());
175     QPushButton *buttonOk = new QPushButton("Ok");
176     QPushButton *buttonCancel = new QPushButton("Cancel");
177
178     QSpinBox *sizeBox = new QSpinBox(askFont);
179     sizeBox->setMinimum(1);
180     sizeBox->setValue(currentFont.pointSize());
181
182     askFont->layout()->addWidget(fontBox);
183     askFont->layout()->addWidget(sizeBox);
184     askFont->layout()->addWidget(buttonOk);
185     askFont->layout()->addWidget(buttonCancel);
186     connect(buttonOk, &QPushButton::clicked, this, [&currentFont, &fontBox,
187         &sizeBox, askFont]()
188     {
189         currentFont.setStyleHint((QFont::StyleHint)fontBox->
currentIndex());
190         askFont->accept();
191         currentFont.setPointSize(sizeBox->value());
192     });
193     connect(buttonCancel, &QPushButton::clicked, this, [askFont]()
194     {
195         askFont->reject();
196     });

```

```

196
197     askFont->exec();
198
199     area->document()->setDefaultFont(currentFont);
200
201     #ifdef EXTRA_VERBOSE
202         std::cerr << currentFont.styleHint() << "!\n";
203     #endif
204 }
205
206 void MainWindow::exitApp()
207 {
208     #ifdef EXTRA_VERBOSE
209         std::cerr << "Exiting the application!\n";
210     #endif
211
212     exit(0);
213 }
214
215 void MainWindow::quit()
216 {
217     #ifdef EXTRA_VERBOSE
218         std::cerr << "Quit Dialog!\n";
219     #endif
220
221     QMessageBox::StandardButton reply = QMessageBox::question(this, "Exit",
222         "Close without saving?",
223         QMessageBox
224         ::Save |
225         QMessageBox
226         ::Cancel |
227         QMessageBox
228         ::Close );
229     if (reply == QMessageBox::Save) {
230         saveFile();
231         exitApp();
232     } else if (reply == QMessageBox::Close) {
233         exitApp();
234     }
235 }
236
237 void MainWindow::setDefaultFilename()
238 {
239     int num = 1;
240     std::ifstream file;
241     do {
242         file.close();
243         filename = DEFAULT_FILENAME + QString::number(num) + "." +
244             DEFAULT_FILETYPE;
245         file.open(filename.toStdString(), std::ifstream::in);
246         ++num;
247     } while (file.good());
248 }

```



```

244
245 void MainWindow::saveFile()
246 {
247     QString text = area->toPlainText();
248
249     if (filename.isEmpty()) {
250         setDefaultFilename();
251     }
252     lastSaveTimestamp = std::time(0);
253     statusBarUpdate();
254     unsavedChanges = false;
255     titleUpdate();
256
257     #ifdef EXTRA_VERBOSE
258         std::cerr << "Saving File to " << filename.toStdString() << "!\n";
259     #endif
260
261     QFile fout(filename);
262     fout.open(QIODevice::ReadWrite | QIODevice::Text);
263     fout.write(text.toUtf8());
264     fout.write("\n");
265     fout.close();
266 }
267
268 void MainWindow::saveAsFile()
269 {
270     filename = QFileDialog::getSaveFileName(this, "Save file as", ".");
271     lastSaveTimestamp = std::time(0);
272     statusBarUpdate();
273     unsavedChanges = false;
274     titleUpdate();
275
276     #ifdef EXTRA_VERBOSE
277         std::cerr << "Saving File As to " << filename.toStdString() << "!\n";
278     #endif
279
280     saveFile();
281 }
282
283 void MainWindow::openFile()
284 {
285     filename = QFileDialog::getOpenFileName(this, "Open", ".");
286     lastSaveTimestamp = std::time(0);
287     statusBarUpdate();
288     unsavedChanges = false;
289     titleUpdate();
290
291     #ifdef EXTRA_VERBOSE
292         std::cerr << "Opening File " << filename.toStdString() << "!\n";
293     #endif
294
295     QFile file(filename);

```

```

296     if (file.open(QIODevice::ReadWrite | QIODevice::Text)) {
297         area->setPlainText(file.readAll());
298         file.close();
299     }
300 }
301
302 void MainWindow::newFile()
303 {
304
305     setDefaultFilename();
306     lastSaveTimestamp = std::time(0);
307     statusBarUpdate();
308     unsavedChanges = false;
309     titleUpdate();
310
311     #ifdef EXTRA_VERBOSE
312         std::cerr << "Creating new File " << filename.toStdString() << " !\n";
313     #endif
314 }

```

Приложение Г

Г.1 Файл SyntaxHighlighter.hpp

```
1 #pragma once
2
3 #include <QSyntaxHighlighter>
4 #include <QString>
5 #include <QTextDocument>
6 #include <QRegularExpression>
7
8 #include <iostream>
9
10 static const size_t MAX_FILETYPE_LEN = 100;
11 static const char SUPPORTED_THEMES[][MAX_FILETYPE_LEN] = {"monokai", "
    tomorrow", "tomorrowNight", "solarized"};
12
13 enum class HighlightingSetup {
14     none = 0,
15     cpp99,
16     cpp14,
17     cpp20,
18     c11,
19     c18,
20     CNT,
21 };
22
23 static const char HighlightingSetups[][MAX_FILETYPE_LEN] = {
24     "none",
25     "C++99",
26     "C++14",
27     "C++20",
28     "C11",
29     "C18",
30 };
31
32 class SyntaxHighlighter : QSyntaxHighlighter {
33 public:
34     SyntaxHighlighter(QTextDocument *parent, HighlightingSetup setup,
35         QString theme);
36
37     void setupSyntaxHighlighter(HighlightingSetup setup, QString theme);
38
39     void setSyntax(HighlightingSetup setup);
40     void setTheme (HighlightingSetup setup, QString theme);
41
42     void setLangRules();
43
44     void setupKeywordPatterns();
45
46     void setColorValues(QString theme);
47
48     void highlightBlock(const QString &text);
```

```

48
49
50 private:
51     HighlightingSetup setup;
52     QString theme;
53
54     QRegularExpression commentStartExpression, commentEndExpression;
55     bool commonTextColorIsWhite = 1;
56
57     QStringList difKeywordPatterns;
58
59     struct HighlightingRule
60     {
61         QRegularExpression pattern;
62         QTextCharFormat format;
63     };
64
65     QVector<HighlightingRule> highlightingRules;
66
67     QTextCharFormat keywordFormat;
68     QTextCharFormat classFormat;
69     QTextCharFormat singleLineCommentFormat;
70     QTextCharFormat multiLineCommentFormat;
71     QTextCharFormat quotationFormat;
72     QTextCharFormat functionFormat;
73     QTextCharFormat keywordPatterns;
74     QTextCharFormat keywordPatterns_C;
75     QTextCharFormat keywordPatterns_Python;
76     QTextCharFormat headerFileFormat;
77     QTextCharFormat numberFormat;
78     QTextCharFormat formatStringFormat;
79     QTextCharFormat operatorFormat;
80     QTextCharFormat phpVarFormat;
81     QTextCharFormat rubyVarFormat;
82     QTextCharFormat tagFormat;
83     QTextCharFormat valueFormat;
84     QTextCharFormat attributeFormat;
85     QTextCharFormat idFormat;
86
87     QColor keywordColor;
88     QColor keyword2Color;
89     QColor functionsColor;
90     QColor valueColor;
91     QColor numColor;
92     QColor operatorColor;
93     QColor formatStringColor;
94     QColor commentColor;
95     QColor varColor;
96     QColor tagColor;
97     QColor htmlAttributesColor;
98     QColor cssClassesIDsColor;
99     QColor cssAttributesColor;
100 };

```

Г.2 Файл SyntaxHighlighter.cpp

[illegible]

```

bdouble\\b"
51      << "\\bstruct\\b" << "\\benum\\b" << "\\bvoid\\b" << "
\\bshort\\b"
52      << "\\blong\\b" << "\\btrue\\b" << "\\bfalse\\b" << "\\
bboolean\\b"
53      << "\\bnull\\b" << "\\bthis\\b" << "\\bfinal\\b"
54      << "\\band\\b" << "\\bor\\b" << "\\bxor\\b"
55      << "\\bconst\\b" << "\\bstatic\\b" << "\\bsigned\\b" << "
"\\bunsigned\\b"
56      << "\\bimport\\b" << "\\bnamespace\\b" << "\\breturn\\b
" << "\\busing\\b"
57      << "\\bfor\\b" << "\\bwhile\\b" << "\\bif\\b" << "\\
belse\\b"
58      << "\\bcase\\b" << "\\bswitch\\b" << "\\bdo\\b" << "\\
bunion\\b"
59      << "\\bnew\\b" << "\\bclass\\b" << "\\bprivate\\b" << "
\\bprotected\\b"
60      << "\\bpublic\\b" << "\\bvirtual\\b" << "\\bslots\\b"
<< "\\bvolatile\\b"
61      << "\\babstract\\b" << "\\bextends\\b" << "\\
bimplements\\b" << "\\bsuper\\b"
62      << "\\btemplate\\b" << "\\btypedef\\b" << "\\btypename
\\b"
63      << "\\btry\\b" << "\\bcatch\\b" << "\\bthrow\\b" << "\\
bbreak\\b";
64      for (const QString &pattern : keywordPatterns) {
65          rule.pattern = QRegularExpression(pattern);
66          rule.format = keywordFormat;
67          highlightingRules.append(rule);
68      }
69
70      numberFormat.setForeground(numColor);
71      rule.pattern = QRegularExpression("\\b[0-9\\.]+\\b");
72      rule.format = numberFormat;
73      highlightingRules.append(rule);
74
75      quotationFormat.setForeground(valueColor);
76      rule.pattern = QRegularExpression("\\\".*\\\"");
77      rule.format = quotationFormat;
78      highlightingRules.append(rule);
79
80      quotationFormat.setForeground(valueColor);
81      rule.pattern = QRegularExpression("'.*'");
82      rule.format = quotationFormat;
83      highlightingRules.append(rule);
84
85      formatStringFormat.setForeground(formatStringColor);
86      rule.pattern = QRegularExpression("%[sdifFuoxXeEgGaAcnp]+\\b");
87      rule.format = formatStringFormat;
88      highlightingRules.append(rule);
89
90      headerFileFormat.setForeground(QColor("#ff6d6d"));
91      rule.pattern = QRegularExpression("#include\\.?[<\\\"].*([>\\\"]");

```

```

92     rule.format = headerFileFormat;
93     highlightingRules.append(rule);
94
95     singleLineCommentFormat.setFontItalic(true);
96     singleLineCommentFormat.setForeground(commentColor);
97     rule.pattern = QRegularExpression("//[^\n]*");
98     rule.format = singleLineCommentFormat;
99     highlightingRules.append(rule);
100
101     multiLineCommentFormat.setFontItalic(true);
102     multiLineCommentFormat.setForeground(commentColor);
103
104     commentStartExpression = QRegularExpression("/\\*");
105     commentEndExpression = QRegularExpression("\\*/");
106 }
107
108 void SyntaxHighlighter::setupKeywordPatterns()
109 {
110     #ifdef EXTRA_VERBOSE
111         std::cerr << "setup = " << static_cast<size_t>(setup) << " (" <<
HighlightingSetups[static_cast<size_t>(setup)] << ")\n";
112     #endif
113
114     switch (setup) {
115     case (HighlightingSetup::cpp99):
116         difKeywordPatterns << "\\bchar\\b" << "\\bint\\b" << "\\bfloat\\b"
<< "\\bdouble\\b"
117
118                                     << "\\bstruct\\b" << "\\benum\\b" << "\\bvoid\\b"
<< "\\bshort\\b"
119                                     << "\\blong\\b" << "\\btrue\\b" << "\\bfalse\\b"
<< "\\bboolean\\b"
120                                     << "\\bthis\\b" << "\\bfriend\\b" << "\\b
bconstexpr\\b"
121
122                                     << "\\bconst\\b" << "\\bstatic\\b" << "\\b
bsigned\\b" << "\\bunsigned\\b"
123                                     << "\\bnamespace\\b" << "\\breturn\\b" << "\\b
busing\\b"
124                                     << "\\bfor\\b" << "\\bwhile\\b" << "\\bif\\b"
<< "\\belse\\b"
125                                     << "\\bcase\\b" << "\\bswitch\\b" << "\\bdo\\b"
<< "\\bunion\\b"
126
127                                     << "\\bnew\\b" << "\\bclass\\b" << "\\bprivate
\\b" << "\\bprotected\\b"
128                                     << "\\bpublic\\b" << "\\bvirtual\\b" << "\\b
bextern\\b" << "\\bvolatile\\b"
129                                     << "\\btemplate\\b" << "\\btypedef\\b" << "\\b
btypename\\b"
130                                     << "\\btry\\b" << "\\bcatch\\b" << "\\bthrow\\b"
<< "\\bbreak\\b"
131                                     << "\\bgoto\\b" << "\\bregister\\b" << "\\b

```

```

132 binline\\b"
133
134 << "\\band\\b" << "\\bbitor\\b" << "\\bor\\b"
135 << "\\bxor\\b"
136 << "\\bor_eq\\b" << "\\band_eq\\b" << "\\
137 bbitand\\b" << "\\bcompl\\b"
138 << "\\bxor_eq\\b" << "\\bnot\\b" << "\\bnot_eq
139 \\b"
140
141 << "\\basm\\b" << "\\bauto\\b"
142 << "\\bbool\\b" << "\\bcontinue\\b"
143 << "\\bdefault\\b" << "\\bdelete\\b" << "\\
144 bdynamic_cast\\b"
145 << "\\bexplicit\\b" << "\\bexport\\b" << "\\
146 bmutable\\b"
147 << "\\boperator\\b" << "\\breinterpret_cast\\b"
148 << "\\btypeid\\b"
149 << "\\bstatic_cast\\b" << "\\bwchar_t\\b" << "
150 \\bfinal\\b" << "\\boverride\\b";
151 break;
152
153 case (HighlightingSetup::cpp14):
154     difKeywordPatterns << "\\bchar\\b" << "\\bint\\b" << "\\bfloat\\b"
155     << "\\bdouble\\b"
156     << "\\bstruct\\b" << "\\benum\\b" << "\\bvoid\\
157 b" << "\\bshort\\b"
158     << "\\blong\\b" << "\\btrue\\b" << "\\bfalse\\b
159 " << "\\bboolean\\b"
160     << "\\bnullptr\\b" << "\\bthis\\b" << "\\
161 bfriend\\b" << "\\bconstexpr\\b"
162     << "\\bconst\\b" << "\\bstatic\\b" << "\\
163 bsigned\\b" << "\\bunsigned\\b"
164     << "\\bnamespace\\b" << "\\breturn\\b" << "\\
165 busing\\b"
166     << "\\bfor\\b" << "\\bwhile\\b" << "\\bif\\b"
167 << "\\belse\\b"
168     << "\\bcase\\b" << "\\bswitch\\b" << "\\bdo\\b"
169 << "\\bunion\\b"
170
171     << "\\bnew\\b" << "\\bclass\\b" << "\\bprivate
172 \\b" << "\\bprotected\\b"
173     << "\\bpublic\\b" << "\\bvirtual\\b" << "\\
174 bextern\\b" << "\\bvolatile\\b"
175     << "\\btemplate\\b" << "\\btypedef\\b" << "\\
176 btypename\\b"
177     << "\\btry\\b" << "\\bcatch\\b" << "\\bthrow\\b
178 " << "\\bbreak\\b"
179     << "\\bgoto\\b" << "\\binline\\b" << "\\
180 bthread_local\\b"
181
182 << "\\band\\b" << "\\bbitor\\b" << "\\bor\\b"
183 << "\\bxor\\b"

```



```

163         << "\\bor_eq\\b" << "\\band_eq\\b" << "\\
bbitand\\b" << "\\bcompl\\b"
164         << "\\bxor_eq\\b" << "\\bnot\\b" << "\\bnot_eq
\\b"

165         << "\\balignas\\b" << "\\balignof\\b" << "\\
basm\\b" << "\\bauto\\b"
166         << "\\bbool\\b" << "\\bchar16_t\\b" << "\\
bchar32_t\\b" << "\\bcontinue\\b"
167         << "\\bdecltype\\b" << "\\bdefault\\b" << "\\
bdelete\\b" << "\\bdynamic_cast\\b"
168         << "\\bexplicit\\b" << "\\bexport\\b" << "\\
bmutable\\b" << "\\bnoexcept\\b"
169         << "\\boperator\\b" << "\\breinterpret_cast\\b"
170         << "\\btypeid\\b" << "\\bstatic_assert\\b"
171         << "\\bstatic_cast\\b" << "\\bwchar_t\\b" << "
\\bfinal\\b" << "\\boverride\\b"
172         << "\\bint8_t\\b" << "\\bint16_t\\b" << "\\
bint32_t\\b" << "\\bint64_t\\b"
173         << "\\bint_fast8_t\\b" << "\\bint_fast16_t\\b"
<< "\\bint_fast32_t\\b" << "\\bint_fast64_t\\b"
174         << "\\bint_least8_t\\b" << "\\bint_least16_t\\b
" << "\\bint_least32_t\\b" << "\\bint_least64_t\\b"
175         << "\\buint8_t\\b" << "\\buint16_t\\b" << "\\
buint32_t\\b" << "\\buint64_t\\b"
176         << "\\buint_fast8_t\\b" << "\\buint_fast16_t\\b
" << "\\buint_fast32_t\\b" << "\\buint_fast64_t\\b"
177         << "\\buint_least8_t\\b" << "\\buint_least16_t
\\b" << "\\buint_least32_t\\b" << "\\buint_least64_t\\b"
178         << "\\bintmax_t\\b" << "\\bwintptr_t\\b" << "\\
buintmax_t\\b" << "\\buintptr_t\\b";
179     break;
180
181     case (HighlightingSetup::cpp20):
182         difKeywordPatterns << "\\bchar\\b" << "\\bint\\b" << "\\bfloat\\b"
<< "\\bdouble\\b"
183         << "\\bstruct\\b" << "\\benum\\b" << "\\bvoid\\
b" << "\\bshort\\b"
184         << "\\blong\\b" << "\\btrue\\b" << "\\bfalse\\b
" << "\\bboolean\\b"
185         << "\\bnullptr\\b" << "\\bthis\\b" << "\\
bfriend\\b" << "\\bconstexpr\\b"
186         << "\\bconst\\b" << "\\bstatic\\b" << "\\
bsigned\\b" << "\\bunsigned\\b"
187         << "\\bnamespace\\b" << "\\breturn\\b" << "\\
busing\\b"
188         << "\\bfor\\b" << "\\bwhile\\b" << "\\bif\\b"
189         << "\\belse\\b"
190         << "\\bcase\\b" << "\\bswitch\\b" << "\\bdo\\b"
<< "\\bunion\\b"
191
192         << "\\bnew\\b" << "\\bclass\\b" << "\\bprivate

```

```

193     \b" << "\\bprotected\b"
194         << "\\bpublic\b" << "\\bvirtual\b" << "\\
bextern\b" << "\\bvolatile\b"
194         << "\\btemplate\b" << "\\btypedef\b" << "\\
btypename\b"
195         << "\\btry\b" << "\\bcatch\b" << "\\bthrow\b
" << "\\bbreak\b"
196         << "\\bgoto\b" << "\\binline\b" << "\\
bthread_local\b"
197
198         << "\\balignas\b" << "\\balignof\b" << "\\
basmb\b" << "\\bauto\b"
199         << "\\bbool\b" << "\\bchar16_t\b" << "\\
bchar32_t\b" << "\\bcontinue\b"
200         << "\\bdecltype\b" << "\\bdefault\b" << "\\
bdelete\b" << "\\bdynamic_cast\b"
201         << "\\bexplicit\b" << "\\bexport\b" << "\\
bmutable\b" << "\\bnoexcept\b"
202         << "\\boperator\b" << "\\breinterpret_cast\b"
    << "\\btypeid\b" << "\\bstatic_assert\b"
203         << "\\bstatic_cast\b" << "\\bwchar_t\b" << "
\\bfinal\b" << "\\boverride\b"
204
205         << "\\band\b" << "\\bbitor\b" << "\\bor\b"
    << "\\bxor\b"
206         << "\\bor_eq\b" << "\\band_eq\b" << "\\
bbitand\b" << "\\bcompl\b"
207         << "\\bxor_eq\b" << "\\bnot\b" << "\\bnot_eq
\b"
208
209         // C++20:
210         << "\\bchar8_t\b" << "\\bconcept\b" << "\\
bconsteval\b" << "\\bconstinit\b"
211         << "\\bco_await\b" << "\\bco_return\b" << "
\\bco_yield\b" << "\\brequires\b"
212         << "\\bstatic_cast\b" << "\\bwchar_t\b" << "
\\bfinal\b" << "\\boverride\b"
213         << "\\bstatic_cast\b" << "\\bwchar_t\b" << "
\\bfinal\b" << "\\boverride\b";
214         break;
215
216     case (HighlightingSetup::c11):
217         difKeywordPatterns << "\\bchar\b" << "\\bint\b" << "\\bfloat\b"
    << "\\bdouble\b"
218         << "\\bstruct\b" << "\\benum\b" << "\\bvoid\b
" << "\\bshort\b"
219         << "\\blong\b"
220         << "\\bconst\b" << "\\bstatic\b" << "\\
bsigned\b" << "\\bunsigned\b"
221         << "\\breturn\b"
222         << "\\bfor\b" << "\\bwhile\b" << "\\bif\b"
    << "\\belse\b"
223         << "\\bcase\b" << "\\bswitch\b" << "\\bdo\b"

```

```

224         << "\\bunion\\b"
                << "\\bvolatile\\b" << "\\bextern\\b" << "\\b
bgoto\\b" << "\\bregister\\b"
225         << "\\btypedef\\b" << "\\bsizeof\\b" << "\\b
brestrict\\b" << "\\binline\\b"
226         << "\\band\\b" << "\\bbitor\\b" << "\\bor\\b"
<< "\\bxor\\b"
227         << "\\bor_eq\\b" << "\\band_eq\\b" << "\\b
bbitand\\b" << "\\bcompl\\b"
228         << "\\bxor_eq\\b" << "\\bnot\\b" << "\\bnot_eq
\\b"
229         // from C99:
230         << "\\brestrict\\b" << "\\binline\\b"
231         << "\\b_Complex\\b" << "\\b_Bool\\b" << "\\b
b_Imaginary\\b"
232
233         << "\\bint8_t\\b" << "\\bint16_t\\b" << "\\b
bint32_t\\b" << "\\bint64_t\\b"
234         << "\\bint_fast8_t\\b" << "\\bint_fast16_t\\b"
<< "\\bint_fast32_t\\b" << "\\bint_fast64_t\\b"
235         << "\\bint_least8_t\\b" << "\\bint_least16_t\\b
" << "\\bint_least32_t\\b" << "\\bint_least64_t\\b"
236         << "\\buint8_t\\b" << "\\buint16_t\\b" << "\\b
buint32_t\\b" << "\\buint64_t\\b"
237         << "\\buint_fast8_t\\b" << "\\buint_fast16_t\\b
" << "\\buint_fast32_t\\b" << "\\buint_fast64_t\\b"
238         << "\\buint_least8_t\\b" << "\\buint_least16_t
\\b" << "\\buint_least32_t\\b" << "\\buint_least64_t\\b"
239         << "\\bintmax_t\\b" << "\\bwintptr_t\\b" << "\\b
buintmax_t\\b" << "\\buintptr_t\\b";
240         break;
241
242         case (HighlightingSetup::c18):
243             difKeywordPatterns << "\\bchar\\b" << "\\bint\\b" << "\\bfloat\\b"
<< "\\bdouble\\b"
244             << "\\bstruct\\b" << "\\benum\\b" << "\\bvoid\\b
b" << "\\bshort\\b"
245             << "\\blong\\b"
246             << "\\bconst\\b" << "\\bstatic\\b" << "\\b
bsigned\\b" << "\\bunsigned\\b"
247             << "\\breturn\\b"
248
249             << "\\band\\b" << "\\bbitor\\b" << "\\bor\\b"
<< "\\bxor\\b"
250             << "\\bor_eq\\b" << "\\band_eq\\b" << "\\b
bbitand\\b" << "\\bcompl\\b"
251             << "\\bxor_eq\\b" << "\\bnot\\b" << "\\bnot_eq
\\b"
252
253             << "\\bfor\\b" << "\\bwhile\\b" << "\\bif\\b"
<< "\\belse\\b"
254             << "\\bcase\\b" << "\\bswitch\\b" << "\\bdof\\b"
<< "\\bunion\\b"

```

```

255         << "\\bvolatile\\b" << "\\bextern\\b" << "\\
bgoto\\b" << "\\binline\\b"
256         << "\\btypedef\\b" << "\\bsizeof\\b" << "\\
brestrict\\b" << "\\bregister\\b"
257         // from C18:
258         << "\\b_Alignas\\b" << "\\b_Alignof\\b" << "\\
b_Atomic\\b" << "\\b_Bool\\b"
259         << "\\b_Complex\\b" << "\\b_Generic\\b" << "\\
b_Imaginary\\b" << "\\b_Noreturn\\b"
260         << "\\b_Static_assert\\b" << "\\b_Thread_local
\\b";
261     break;
262
263     case (HighlightingSetup::none):
264         break;
265 }
266 }
267
268 void SyntaxHighlighter::setColorValues(QString theme)
269 {
270     if (theme == "monokai"){
271         // monokai
272         commonTextColorIsWhite = 1;
273         keywordColor = QColor(102, 217, 239);
274         keyword2Color = QColor(249, 38, 114);
275         functionsColor = QColor(166, 226, 46);
276         valueColor = QColor(230, 218, 117);
277         numColor = QColor(174, 130, 255);
278         operatorColor = QColor(249, 38, 114);
279         formatStringColor = QColor(174, 130, 255);
280         commentColor = QColor(178, 179, 191);
281         varColor = QColor(102, 217, 239);
282         tagColor = QColor(249, 38, 114);
283         htmlAttributesColor = QColor(166, 226, 46);
284         cssClassesIDsColor = QColor(166, 226, 46);
285         cssAttributesColor = QColor(102, 217, 239);
286
287     } else if (theme == "tomorrow"){
288         // tomorrow
289         commonTextColorIsWhite = 0;
290         keywordColor = QColor(135, 88, 166);
291         keyword2Color = QColor(135, 88, 166);
292         functionsColor = QColor(66, 114, 173);
293         valueColor = QColor(112, 138, 0);
294         numColor = QColor(245, 135, 32);
295         operatorColor = QColor(77, 77, 76);
296         formatStringColor = QColor(199, 40, 40);
297         commentColor = QColor(144, 143, 140);
298         varColor = QColor(199, 40, 40);
299         tagColor = QColor(199, 40, 40);
300         htmlAttributesColor = QColor(245, 135, 32);
301         cssClassesIDsColor = QColor(62, 153, 158);
302         cssAttributesColor = QColor(77, 77, 76);

```

```

303
304     } else if (theme == "tomorrowNight"){
305         // tomorrow night
306         commonTextColorIsWhite = 1;
307         keywordColor = QColor(177, 149, 186);
308         keyword2Color = QColor(177, 149, 186);
309         functionsColor = QColor(128, 162, 189);
310         valueColor = QColor(182, 189, 106);
311         numColor = QColor(222, 146, 95);
312         operatorColor = QColor(197, 199, 198);
313         formatStringColor = QColor(222, 146, 95);
314         commentColor = QColor(149, 150, 149);
315         varColor = QColor(204, 102, 102);
316         tagColor = QColor(204, 102, 102);
317         htmlAttributesColor = QColor(222, 146, 95);
318         cssClassesIDsColor = QColor(138, 189, 181);
319         cssAttributesColor = QColor(197, 199, 198);
320
321     } else if (theme == "solarized") {
322         // Solarized
323         commonTextColorIsWhite = 0;
324         keywordColor = QColor(181, 137, 0);
325         keyword2Color = QColor(133, 153, 0);
326         functionsColor = QColor(88, 110, 117);
327         valueColor = QColor(42, 161, 152);
328         numColor = QColor(42, 161, 152);
329         operatorColor = QColor(181, 137, 0);
330         formatStringColor = QColor(220, 50, 47);
331         commentColor = QColor(178, 179, 191);
332         varColor = QColor(38, 139, 210);
333         tagColor = QColor(38, 139, 210);
334         htmlAttributesColor = QColor(181, 137, 0);
335         cssClassesIDsColor = QColor(133, 153, 0);
336         cssAttributesColor = QColor(77,171,171);
337     } else {
338         std::cerr << "ERROR: unknown theme provided to SyntaxHighlighter!\n
339     ";
340 }
341
342 void SyntaxHighlighter::highlightBlock(const QString &text)
343 {
344     for (const HighlightingRule &rule : qAsConst(highlightingRules)) {
345         QRegularExpressionMatchIterator matchIterator = rule.pattern.
346         globalMatch(text);
347         while (matchIterator.hasNext()) {
348             QRegularExpressionMatch match = matchIterator.next();
349             setFormat(match.capturedStart(), match.capturedLength(), rule.
350             format);
351         }
352     }
353     setCurrentBlockState(0);

```

```

353
354     if (commentStartExpression != QRegularExpression("") &&
commentEndExpression != QRegularExpression("")) {
355         int startIndex = 0;
356         if (previousBlockState() != 1)
357             startIndex = text.indexOf(commentStartExpression);
358
359         while (startIndex >= 0) {
360             QRegularExpressionMatch match = commentEndExpression.match(text
, startIndex);
361             int endIndex = match.capturedStart();
362             int commentLength = 0;
363             if (endIndex == -1) {
364                 setCurrentBlockState(1);
365                 commentLength = text.length() - startIndex;
366             }
367             else
368             {
369                 commentLength = endIndex - startIndex
+ match.capturedLength();
370             }
371             setFormat(startIndex, commentLength, multiLineCommentFormat);
372             startIndex = text.indexOf(commentStartExpression, startIndex +
commentLength);
373         }
374     }
375 }
376 }

```

Приложение Д

Д.1 Файл TextEditor.hpp

```
1 #pragma once
2 #include <QPlainTextEdit>
3 #include <QPainter>
4 #include <QTextBlock>
5
6 class CodeEditor : public QPlainTextEdit
7 {
8     Q_OBJECT
9
10 public:
11     CodeEditor(QWidget *parent = nullptr);
12
13     void lineNumberAreaPaintEvent(QPaintEvent *event);
14     int  lineNumberAreaWidth();
15
16     bool hideLineNumber;
17     void hideLineNumberArea()
18     {
19         hideLineNumber = true;
20     }
21
22     void showLineNumberArea()
23     {
24         hideLineNumber = false;
25     }
26
27 protected:
28     void resizeEvent(QResizeEvent *event) override;
29
30 private slots:
31     void updateLineNumberAreaWidth();
32     void highlightCurrentLine();
33     void updateLineNumberArea(const QRect &rect, int dy);
34
35 private:
36     QWidget *lineNumberArea;
37
38 friend class MainWindow;
39 };
40
41 class LineNumberArea : public QWidget
42 {
43 public:
44     LineNumberArea(CodeEditor *editor) : QWidget(editor), codeEditor(editor)
45     {}
46
47     QSize sizeHint() const override
48     {
```

```

49         return QSize(codeEditor->lineNumberAreaWidth(), 0);
50     }
51
52 protected:
53     void paintEvent(QPaintEvent *event) override
54     {
55         codeEditor->lineNumberAreaPaintEvent(event);
56     }
57
58 private:
59     CodeEditor *codeEditor;
60 };

```

Д.2 Файл TextEditor.cpp

```

1 #include "TextEditor.hpp"
2
3 CodeEditor::CodeEditor(QWidget *parent) : QPlainTextEdit(parent)
4 {
5     hideLineNumber = false;
6     lineNumberArea = new LineNumberArea(this);
7
8     connect(this, &CodeEditor::blockCountChanged,      this, &CodeEditor::
updateLineNumberAreaWidth);
9     connect(this, &CodeEditor::updateRequest,          this, &CodeEditor::
updateLineNumberArea);
10    connect(this, &CodeEditor::cursorPositionChanged, this, &CodeEditor::
highlightCurrentLine);
11
12    updateLineNumberAreaWidth();
13    highlightCurrentLine();
14 }
15
16 int CodeEditor::lineNumberAreaWidth()
17 {
18     if (hideLineNumber)
19         return 0;
20
21     int digits = 1;
22     int max = qMax(1, blockCount());
23     while (max >= 10) {
24         max /= 10;
25         ++digits;
26     }
27
28     int space = 3 + fontMetrics().horizontalAdvance(QLatin1Char('9')) *
digits;
29
30     return space;
31 }
32
33 void CodeEditor::updateLineNumberAreaWidth()
34 {
35     setViewportMargins(lineNumberAreaWidth(), 0, 0, 0);

```



```

36 }
37
38 void CodeEditor::updateLineNumberArea(const QRect &rect, int dy)
39 {
40     if (dy)
41         lineNumberArea->scroll(0, dy);
42     else
43         lineNumberArea->update(0, rect.y(), lineNumberArea->width(), rect.
height());
44
45     if (rect.contains(viewport()->rect()))
46         updateLineNumberAreaWidth();
47 }
48
49 void CodeEditor::resizeEvent(QResizeEvent *e)
50 {
51     QPlainTextEdit::resizeEvent(e);
52
53     QRect cr = contentsRect();
54     lineNumberArea->setGeometry(QRect(cr.left(), cr.top(),
lineNumberAreaWidth(), cr.height()));
55 }
56
57 void CodeEditor::highlightCurrentLine()
58 {
59     QList<QTextEdit::ExtraSelection> extraSelections;
60
61     if (!isReadOnly()) {
62         QTextEdit::ExtraSelection selection;
63
64         QColor lineColor = QColor(Qt::yellow).lighter(160);
65
66         selection.format.setBackground(lineColor);
67         selection.format.setProperty(QTextFormat::FullWidthSelection, true)
;
68         selection.cursor = textCursor();
69         selection.cursor.clearSelection();
70         extraSelections.append(selection);
71     }
72
73     setExtraSelections(extraSelections);
74 }
75
76 void CodeEditor::lineNumberAreaPaintEvent(QPaintEvent *event)
77 {
78     QPainter painter(lineNumberArea);
79     painter.fillRect(event->rect(), Qt::lightGray);
80     QTextBlock block = firstVisibleBlock();
81     int blockNumber = block.blockNumber();
82     int top = qRound(blockBoundingGeometry(block).translated(
contentOffset()).top());
83     int bottom = qRound(blockBoundingRect(block).height()) + top;
84     while (block.isValid() && top <= event->rect().bottom()) {

```

```

85         if (block.isVisible() && bottom >= event->rect().top()) {
86             QString number = QString::number(blockNumber + 1);
87             painter.setPen(Qt::black);
88             painter.drawText(0, top, lineNumberArea->width(), fontMetrics()
.height(),
89                             Qt::AlignRight, number);
90         }
91
92         block = block.next();
93         top = bottom;
94         bottom = qRound(blockBoundingRect(block).height()) + top;
95         ++blockNumber;
96     }
97 }

```