Лабораторная работа № 6

Сборка простейшего приложения с использованием библиотеки Qt5 в операционной системе Linux

Цель работы

Познакомиться с библиотекой Qt (кросс-платформенный инструментарий разработки ПО на языке C++) и этапами сборки проекта, использующего Qt, с помощью утилиты qmake. Совершенствовать навык хранения исходного кода проекта в репозитории.

Работа заключается в последовательном создании трех приложений, каждый из которых представляет собой усложнение предыдущего. Исходный код каждого из приложений в процессе работы должен быть сохранен в репозитории (в локальном и удаленном). Работа выполняется в режиме терминала.

Задание

Создание простейшего приложения - пустое окно

1. Создать в Gitea новый репозиторий - каталог проекта (например lab6), получить его локальную копию.
2. Перейти в локальную копию репозитория, открыть любой простейший текстовый редактор, ввести приведённый ниже текст и сохранить его с именем файла main.cpp:

```
#include <QtWidgets>

int main(int argc, char *argv[])
{
   QApplication app(argc, argv);
   QMainWindow *mw=new QMainWindow(0, Qt::Window);
   mw->setWindowTitle("Hello, Qt5");
   mw->resize(400, 300);
   mw->show();
   return app.exec();
}
```

3. Определить назначение каждой строки программы.
4. В терминале перейти в каталог проекта lab6, просмотреть содержимое этого каталога. Просмотреть статус репозитория. Настроить свойства каталога проекта и входящих в него файлов так, чтобы в репозитории хранились только файлы с исходным кодом.
5. Выполнить в терминале компиляцию и сборку проекта при помощи утилит qmake и make:
   - выполнить команду **qmake -project**, при этом в нашей папке

создается файл проекта lab6.pro (pro-файл),
- откройте файл lab6.pro в простейшем текстовом редакторе и добавьте строку текста **QT += widgets** после строки INCLUDEPATH += .
- выполнить команду **qmake lab6.pro** или **qmake**, при этом в нашей папке создается файл Makefile,
- выполнить команду **make** - производим компиляцию; если программа написана правильно и терминал не выдает отчета об ошибках, то в нашей папке появляется приложение под именем lab6. Запускаем его и видим пустое окно с подписью Hello, Qt5.

6. Сохранить файл с исходным кодом в репозитории.

Простейшее приложение - окно с кнопкой

7. Отредактировать исходний код в файле main.cpp:

```
#include <QtWidgets>

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    QWidget window;
    window.resize(200, 120);
    window.setWindowTitle("Button");
    QPushButton quit("Quit", &window);
    quit.setFont(QFont("Times", 18, QFont::Bold));
    quit.setGeometry(10, 40, 180, 40);
    QObject::connect(&quit,    SIGNAL(clicked()),    &app,
SLOT(quit()));

    window.show();
    return app.exec();
}
```

8. Выполнить сборку проекта и проверить работу программы.
9. Поместить исходный код в репозиторий.

Приложение - окно с кнопками и окном редактора Edit

10. Отредактировать проект: добавить модуль класса окна, после чего в проекте будут три файла mywidget.h, mywidget.cpp, main.cpp. Тексты файлов приведены ниже.

//main.cpp

```
//
#include <QtWidgets>
#include "mywidget.h"
```

```
int main (int argc, char *argv[])
{
    QApplication app(argc, argv);
    MyWidget widget;
    widget.show();
    return app.exec();
}
```

//mywidget.h

```
//
#ifndef MYWIDGET_H
#define MYWIDGET_H

#include <QtWidgets>
#include <QString>

class MyWidget : public QMainWindow
{
    Q_OBJECT
private:
    QLineEdit* line1;
    QLineEdit* line2;
    QPushButton* bsqua;

    QPushButton* bclear;
    QLabel* labsign;
    QLabel* labis;

private slots:
    void slotClear();
    void slotSqua();

public:
    MyWidget();
    ~MyWidget();
};
#endif
```

//mywidget.cpp

```
//
#include "mywidget.h"

MyWidget::~MyWidget()
{
}
```

```
void MyWidget::slotSqua()
{
    double d;
    QString tmp;

    d=line1->text().toDouble();
    d=d*d;
    labsign->setText("");
    line2->setText(tmp.setNum(d));
}

void MyWidget::slotClear()
{
    line1 -> setText("");
    line2 -> setText("");
    labsign -> setText("");
}

MyWidget::MyWidget()
{
    this -> resize(300,150);

    line1 = new QLineEdit(this);
    line2 = new QLineEdit(this);

    bsqua = new QPushButton("*",this);
    bclear = new QPushButton("C",this);
    labsign=new QLabel("", this);
    labis = new QLabel("^2=",this);

    line1 -> move(60,30); line1 -> resize(70,20);
    labis -> move(140,30); labis -> resize(30,20);
    line2 -> move(180,30); line2 -> resize(70,20);
    line2 -> setReadOnly(true);
    bsqua -> move(70,80); bsqua -> resize(50,50);
    bclear -> move(190,80); bclear -> resize(50,50);

    connect(bsqua,         SIGNAL(clicked()),         this,
SLOT(slotSqua()));
    connect   (bclear,   SIGNAL   (clicked()),   this,   SLOT
(slotClear()));
}
```

11. Выполнить сборку проекта и проверить работу программы.

12. Поместить исходный код в репозиторий.
13. Просмотреть список коммитов вашего проекта.
14. Дополнительное задание. Включите в ваше приложение функцию обработки строки. Варианты заданий приведены ниже. Исходный код приложения также поместите в репозиторий.

Варианты заданий

№ варианта	Задание
1	Ввод строки и вывод на экран длины строки и 3-го символа
2	Ввод строки, добавление в начале строки символа 0 (ноль), в конце строки – первого символа исходной строки
3	Ввод строки, перевод её в нижний регистр, добавление новой строки к старой и вывод результата на экран
4	Ввод строки, удаление первого символа, вывод на экран получившейся строки и её длины
5	Ввод строки, замену первого символа на последний и последнего на первый
6	Ввод двух строк, вычисление суммы длин этих строк и вывод на экран
7	Ввод двух строк, определение разности длин этих строк и добавление этого значения (в виде подстроки) к первой строке
8	Ввод двух строк, объединение этих строк в одну, вывод на экран получившейся строки и её длины
9	Ввод двух строк, создание третьей строки, состоящей из первых символов введённых строк и вывод её на экран
10	Ввод двух строк, вставка второй строки в первую, начиная с третьего символа, вывод на экран получившейся строки и её длины
11	Ввод строки, ввод номера позиции в строке, вставка первого символа в строке в заданную позицию
12	Ввод двух строк, обмен первыми символами между строками, вывод на экран получившихся строк

Справочный материал

Класс QString

Предназначен для работы со строками Unicode.
Объявление строки:

```
QString str;
QString str1="abcd";
```

Получение символа, стоящего в заданной позиции строки:

```
QChar ch=str1[0];
```

Действия над строками

Присваивание:

```
QString str2;
str2=str1;
```

Добавление стоки:

```
str2+=str1;
```

Сложение (конкатенация, склеивание строк):

```
QString str3=str1+str2;
```

Методы класса QString:

- определение количества символов в строке

```
int length () const
```

- поиск подстроки (первое вхождение)

```
int indexOf (
        const QString& str,
        int from = 0,
        Qt::CaseSensitivity cs = Qt::CaseSensitive
        ) const
```

- поиск подстроки (последнее вхождение, поиск идет с конца строки)

```
int lastIndexOf (
        const QString& str,
        int from = -1,
        Qt::CaseSensitivity cs = Qt::CaseSensitive
            ) const
```

- вставка подстроки в заданную позицию

```
QString& insert(int position, const QString& str)
```

- удаление заданного количества символов

```
QString& remove(int position, int n)
```

- замена подстроки

```
QString& replace(int position,
                 int n, const QString& after)
```

- получение части строки

```
QString mid(int position, int n =  -1) const
```

- преобразование строки в целое число

```
int toInt()
```

- преобразование строки в вещественное число
```
double toDouble()
```

- преобразование символов строки в прописные буквы
```
QString toUpper()
```

- преобразование символов строки в строчные буквы
```
QString toLower()
```

Примеры

Очистка строки («неопределенная» строка):
```
str.clear();
```

Замена регистра символов:
```
QString str = "ABcd23EF";
str = str.toLower();   // str == "abcd23ef"
```