ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ
им. проф. М. А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

А. С. Дюбов

КОМПЬЮТЕРНОЕ ОБЕСПЕЧЕНИЕ РАСЧЕТНО-ПРОЕКТНОЙ И ЭКСПЕРИМЕНТАЛЬНО-ИССЛЕДОВАТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ

**Учебно-методическое пособие
по выполнению курсовой работы**

**САНКТ-ПЕТЕРБУРГ
2017**

УДК 004.42(075.8)
ББК 32.973я73
Д95

Рецензент
кандидат технических наук,
доцент кафедры сетей связи и передачи данных СПбГУТ
И. В. Гришин

Рекомендовано к печати
редакционно-издательским советом СПбГУТ

Дюбов, А. С.

Д95 Компьютерное обеспечение расчетно-проектной и экспериментально-исследовательской деятельности: учебно-методическое пособие по выполнению курсовой работы / А. С. Дюбов ; СПбГУТ. – СПб., 2017. – 32 с.

Даны методические рекомендации к выполнению курсовой работы. Написано в соответствии с программой дисциплины «Компьютерное обеспечение расчетно-проектной и экспериментально-исследовательской деятельности».

Предназначено для студентов, обучающихся по направлению подготовки всех форм обучения 11.03.02 «Инфокоммуникационные технологии и системы связи».

УДК 004.42(075.8)
ББК 32.973я73

Содержание

ВВЕДЕНИЕ

Учебно-методическое пособие предназначено для студентов, выполняющих курсовую работу по дисциплине «Компьютерное обеспечение расчетно-проектной и экспериментально-исследовательской деятельности», являющейся одной из дисциплин профессионального цикла учебного плана подготовки бакалавра по направлению 11.03.02 «Инфокоммуникационные технологии и системы связи». В процессе изучения данной дисциплины студенты должны получить знания о современных средствах вычислительной техники, программных продуктах и тенденциях их развития, приобрести навыки применения современного программного обеспечения для автоматизации задач исследования и проектирования. Курсовая работа является одним из видов промежуточной аттестации.

В курсовой работе студентам предлагается разработать программу на языке C++ с использованием библиотеки Qt и среды разработки Qt Creator 5.2. Методические указания разработаны в предположении, что студенты имеют навыки работы на компьютере и общее представление о языках программирования, однако не знакомы с Qt.

На сегодняшний день Qt широко используется разработчиками во всем мире. Библиотека Qt доступна в исходных текстах, сопровождается хорошо проработанной документацией, предоставляет богатые возможности для разработки графического интерфейса. В комплект Qt входит интегрированная среда разработки, включающая редактор кода, интерактивный отладчик, редактор графического интерфейса, справочную систему. Благодаря этому Qt набирает все бóльшую популярность. Пользователями Qt являются многие известные компании, среди которых AT&T, Canon, HP, Bosch, IBM, Motorola, NEC, Sony, Siemens, Sharp, Xerox. Модули Qt использованы при написании известных программ: Skype, Adobe Photoshop Album, Google Earth и др. [1].

Немало полезной информации можно найти на многочисленных форумах и в статьях, посвященных Qt. Некоторые из них использованы при подготовке данного методического пособия [2–4].

1. ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Целью курсовой работы является развитие навыков разработки прикладного программного обеспечения на языке C++ с использованием интегрированной среды разработки Qt Creator.

Задачей курсовой работы является разработка программы для автоматизации расчета максимально допустимой длины регенерационного участка волоконно-оптической линии связи (ВОЛС) по дисперсии и затуханию.

При выполнении курсовой работы следует выполнить следующие задания:

1) загрузить из сети Интернет и установить интегрированную среду разработки Qt Creator;

2) ознакомиться с интерфейсом и методами работы в среде Qt Creator, выполнить практические задания, разработать учебную программу;

3) ознакомиться с методикой расчета длины регенерационного участка ВОЛС по дисперсии и затуханию;

4) разработать интерфейс программы расчета длины регенерационного участка;

5) разработать алгоритм работы программы;

6) написать исполняемый код на языке C++;

7) разработать руководство пользователя;

8) оформить пояснительную записку к курсовой работе;

9) провести самостоятельную проверку выполненной работы.

В результате курсовой работы должны быть выполнены все пункты задания (1–9), разработана программа для персонального компьютера (ПК), оформлена пояснительная записка к решению. Пояснительная записка должна иметь следующую структуру.

1. Введение.

2. Вариант задания. Исходные данные.

3. Разработка учебной программы в Qt Cteator.

4. Методика расчета длины регенерационного участка ВОЛС.

5. Алгоритм работы программы для расчета длины регенерационного участка ВОЛС.

6. Разработка программы для расчета длины регенерационного участка ВОЛС.

7. Руководство оператора для разработанной программы.

8. Заключение.

9. Список использованных источников.

Для проверки и защиты курсовой работы сдается печатная версия пояснительной записки и электронная версия разработанной программы (программа может быть записана на CD диск и помещена в приложение к пояснительной записке или отправлена по электронной почте руководителю).

1.1. Требования к разрабатываемой программе

Интерфейс программы должен представлять собой одну или несколько форм (окон), на которых расположены элементы управления: поля редактирования, надписи, кнопки для управления и другие необходимые элементы.

Пользователь должен иметь возможность вводить исходные данные и получать значения длины участков регенерации по затуханию и дисперсии.

На форме должны присутствовать: фамилия, имя, номер группы и номер варианта разработчика программы. Эти же данные следует включить как комментарий в начало исполняемого кода.

При написании исходного кода программы рекомендуется выбирать имена переменных и компонентов осмысленно, в соответствии с их назначением и логикой работы. Улучшить читаемость кода можно расстановкой отступов и пробелов. Чтобы облегчить понимание программы, следует сопровождать ее текст комментариями.

1.2. Варианты задания

Рассчитываемая длина регенерационного участка ВОЛС зависит от нескольких параметров. Большинство параметров для расчета должны задаваться пользователем в полях ввода на форме разрабатываемой программы. В зависимости от варианта индивидуального задания некоторые параметры следует считать фиксированными, т. е. не доступными для изменения пользователем. Эти параметры задаются присвоением значений переменным в исходном коде программы.

Следует выполнить расчет длины регенерационного участка для исходных данных, определенных индивидуальным заданием, и привести в пояснительной записке экран программы с результатами расчета.

Индивидуальный вариант задания курсовой работы определяется двумя последними цифрами зачетной книжки или студенческого билета в соответствии с табл. 1 (№ 1 – предпоследняя цифра зачетной книжки) и табл. 2 (№ 2 – последняя цифра зачетной книжки).

Таблица 1

Параметр варианта, определяемый предпоследней цифрой зачетной книжки

№ 1	0	1	2	3	4	5	6	7	8	9
Скорость передачи, C_T, Мбит/с	155	620	2500	10000	155	620	2500	1000	155	620

Таблица 2

Параметр варианта, определяемый последней цифрой зачетной книжки

№ 2	0	1	4	5	8	9	2	3	6	7
Тип оптического волокна	G.652 (SSF)			G.653 (DSF)			G.655 (NZDSF)			

Например, если две последние цифры билета (зачетной книжки) 12, то расчет длины регенерационного участка производится для скорости передачи 620 Мбит/с и волокна G.655 (NZDSF).

1.3. Требования к оформлению пояснительной записки

При оформлении пояснительной записки курсовой работы следует руководствоваться требованиями ГОСТ 7.32–2001 «Отчет о научно-исследовательской работе. Структура и правила оформления» [5].

Пояснительная записка оформляется печатным способом на листах белой бумаги формата А4 (на одной стороне листа). При наборе текста рекомендуется гарнитура шрифта Times New Roman, размер шрифта основного текста – кг. 14, межстрочный интервал – 1,5. Отступы: слева – 25 мм, справа – 10 мм, сверху и снизу – 20 мм. Номера страниц проставляются в центре нижней части листа без точки. Выполненная работа брошюруется.

2. ЗАГРУЗКА И УСТАНОВКА Qt

Для разработки приложений следует использовать интегрированную среду разработки Qt Creator. Дистрибутив среды разработки Qt 5.2.0 для Windows 32-bit (MinGW 4.8, OpenGL, 689 MB) можно загрузить на сайте «Qt Project» по адресу:

URL*: http://qt-project.org/downloads.*

Документация и примеры программ доступны в разделе «Documentation» по адресу:

URL: *http://qt-project.org/doc/.*

Для установки среды разработки Qt Creator 5.2 и необходимых библиотек для операционной системы Windows следует загрузить на компьютер и запустить файл:

qt-windows-opensource-5.2.0-mingw48_opengl-x86-offline.exe,

затем несколько раз нажать кнопку «Далее». На рис. 1 и 2 показаны окна программы в процессе установки.

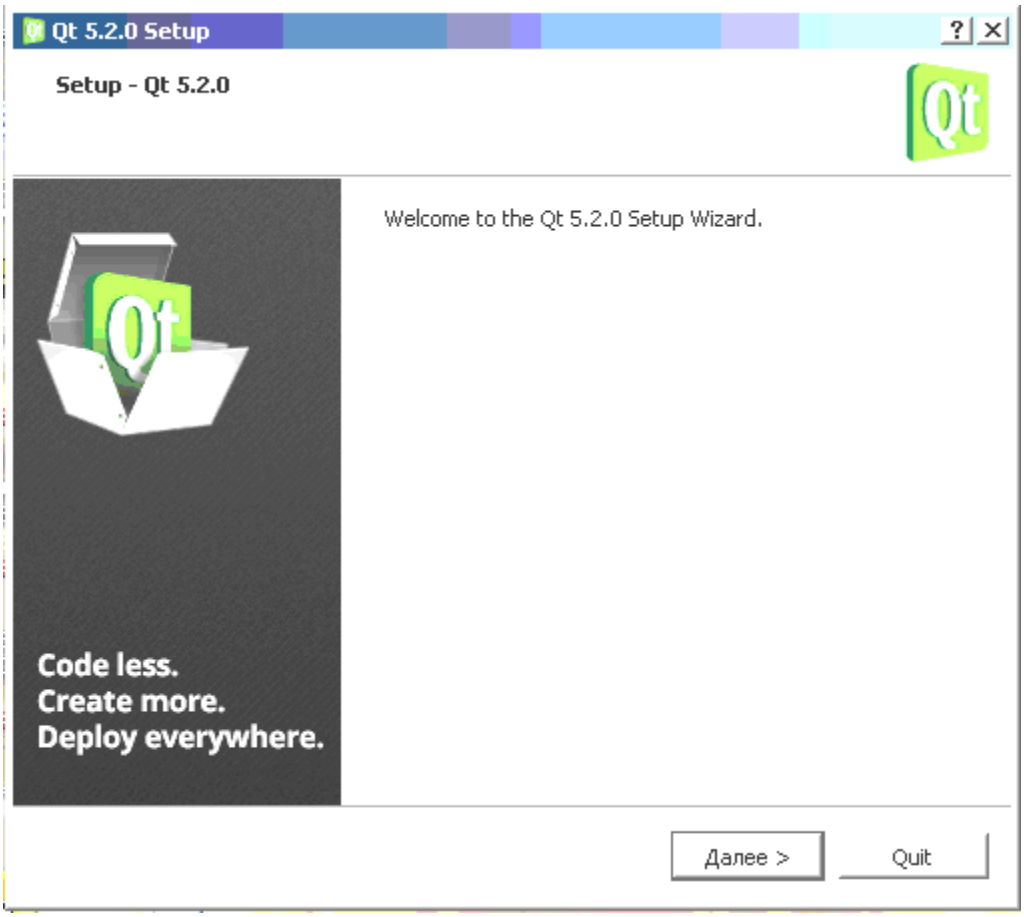

Рис. 1. Экран программы установки Qt

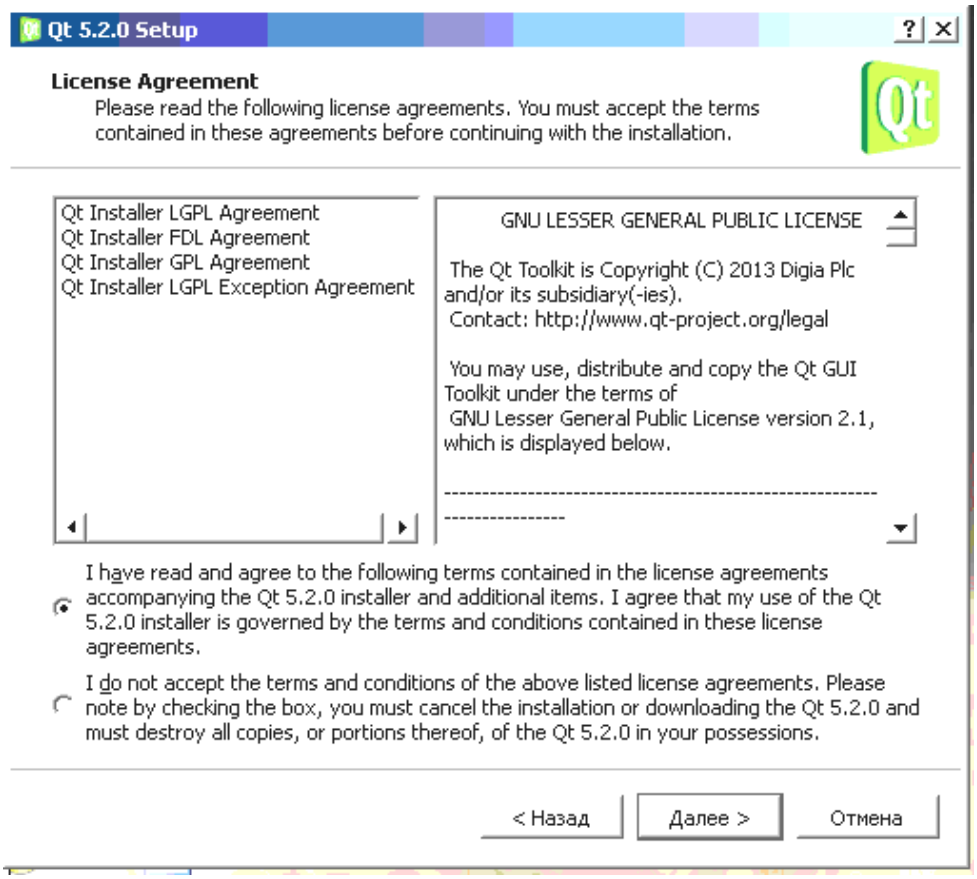

Рис. 2. Экран программы установки Qt. Выбор лицензии

На рис. 3 показано окно среды разработки (Integrated Development Environment – IDE) при первом запуске.

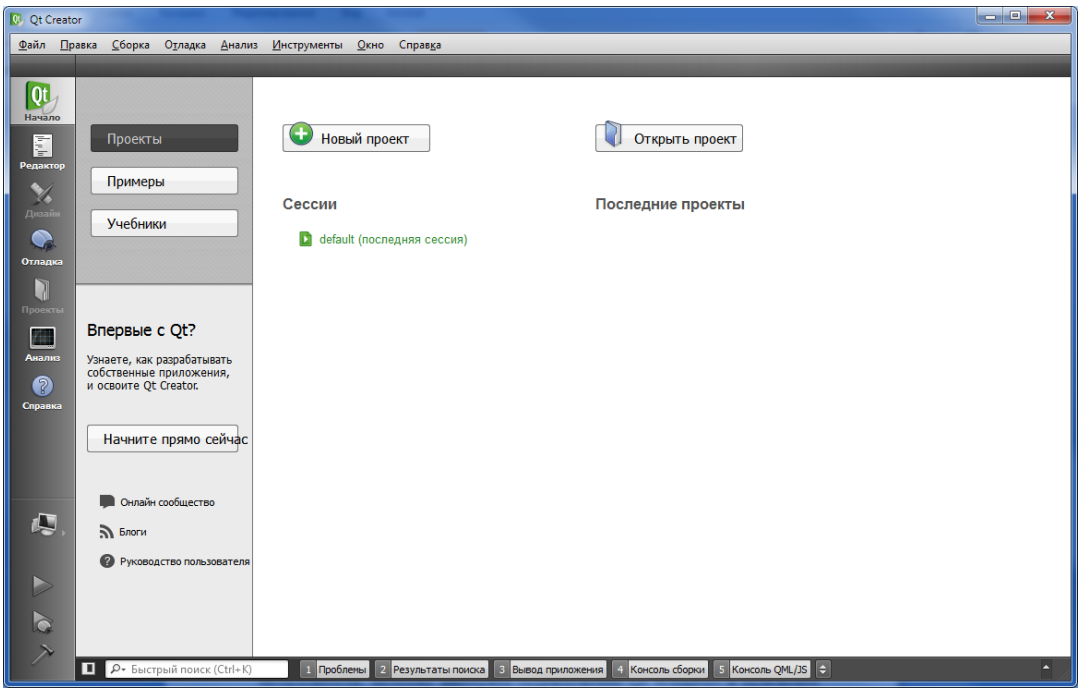

Рис. 3. Окно «Начало» Qt Creator при перовом запуске

3. РАЗРАБОТКА ПРОГРАММЫ

В данном разделе приводится описание интерфейса интегрированной среды разработки Qt Creator. Для получения навыков работы с Qt предлагается выполнить несколько заданий и создать учебную программу. Учебная программа будет состоять из одной формы (главное окно программы), на которой расположены: два поля редактирования, три кнопки и надписи. Учебная программа, описываемая в данном разделе, не рассчитывает длину регенерационного участка, а служит для демонстрации приемов работы с Qt.

Экран разработанной учебной программы (PrintScreen) следует привести в пояснительной записке к курсовой работе, что послужит подтверждением выполнения данного задания.

3.1. Интерфейс Qt Creator. Создание проекта

Создайте новый проект. Для создания нового проекта нажмите кнопку «Новый проект» (рис. 3) и задайте параметры нового проекта. Необходимо создать приложение с графическим интерфейсом пользователя, поэтому в появившемся окне «Новый проект» выберите шаблон «Приложение Qt Widgets» и нажмите кнопку «Выбрать…».

В следующем появившемся окне, показанном на рис. 4, задайте название проекта, например «*first*». Укажите путь для размещения проекта (можно оставить по умолчанию). В пути размещения следует использовать только латинские символы. Использование кириллицы при задании пути размещения проекта приводит к ошибке компиляции.

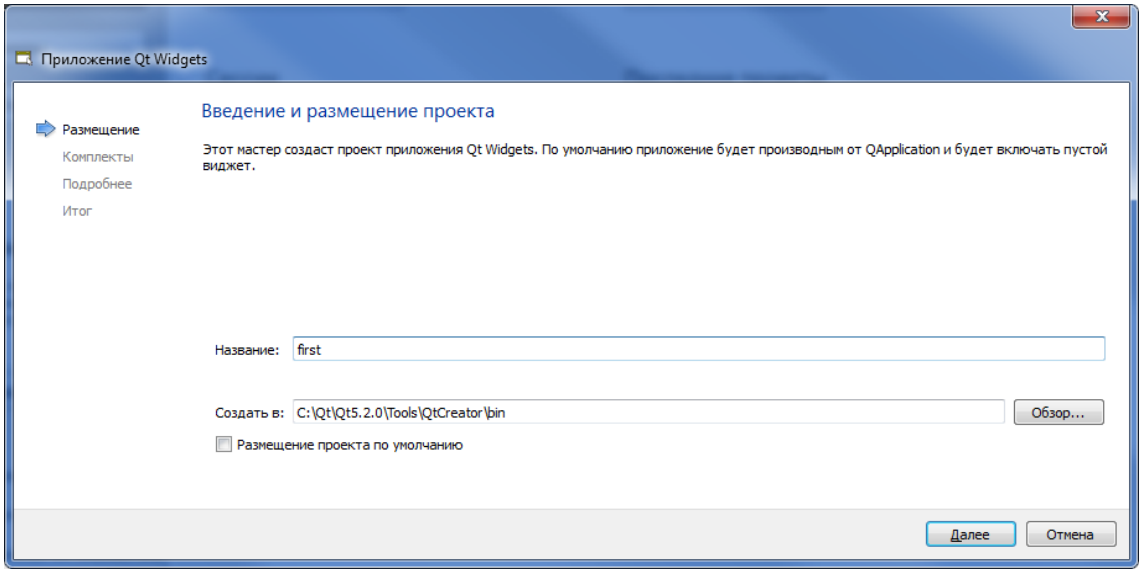

Рис. 4. Задание ввода имени и размещения проекта

Остальные параметры, предлагаемые по умолчанию, оставьте без изменений, несколько раз нажмите кнопку «Далее» и затем нажмите кнопку «Завершить». В результате должно открыться окно среды разработки, показанное на рис. 5.

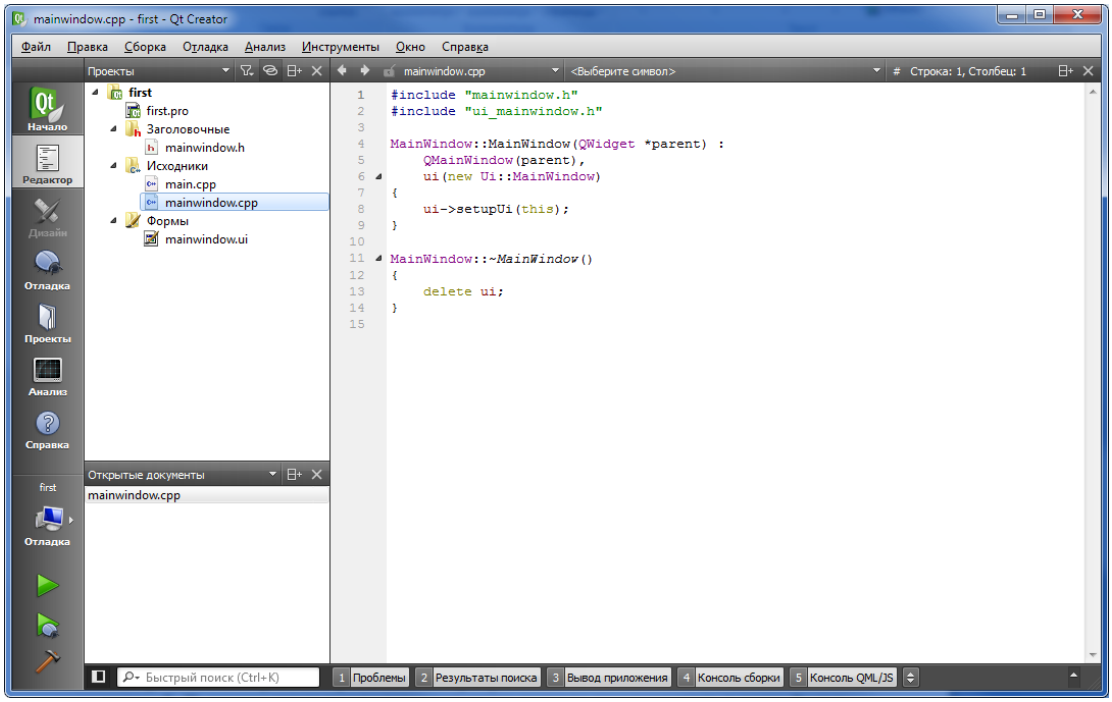

Рис. 5. Окно интегрированной среды разработки. Первый проект

Рассмотрим компоненты графического интерфейса среды разработки:

1) в самом верху расположено главное меню, свойственное многим приложениям Windows;

2) с левой стороны находится полоса смены режимов работы: редактирование, отладка, проекты, справка, а также кнопки компилирования и запуска;

3) правее расположена боковая панель, в которой перечислены файлы, составляющие проект:

файл проекта: first.pro;

заголовочный файл: mainwindow.h;

исполняемые файлы: main.cpp, mainwindow.cpp;

файл формы: mainwindow.ui.

Если навести мышку на имя файла, то всплывающая подсказка покажет путь, где сохранен файл. Двойной клик мышкой открывает файл для редактирования;

4) самое большое окно среды разработки – окно редактора. В нем можно редактировать код программы;

5) в самом низу располагается окно вывода приложения, в него будут выводиться сообщения о ходе компиляции и сообщения об ошибках (если будут ошибки).

Сохраните проект. Для этого в главном меню выберете команду (Файл) => (Сохранить все).

Запустите созданную программу. Для этого нажмите кнопку с зеленым треугольником.

Сейчас программа представляет собой пустое окно «MainWindow». В дальнейшем мы будем добавлять элементы интерфейса и развивать только что созданный проект. Сейчас его можно закрыть. Продолжить работу над проектом можно, запустив Qt Creator и выбрав ссылку «*first*» в списке «Последние проекты».

Итак, мы успешно установили Qt и создали проект (приложение, состоящее из одной формы), запустили его и убедились, что все необходимое установлено правильно.

3.2. Добавление виджетов на форму и редактирование их свойств

Усовершенствуем главное окно учебной программы. Для этого разместим на форме элементы управления – виджеты (Widget) и визуальные объекты – кнопки и поля редактирования.

Используя боковую панель, откройте файл mainwindow.ui в режиме «Дизайн» (рис. 6).

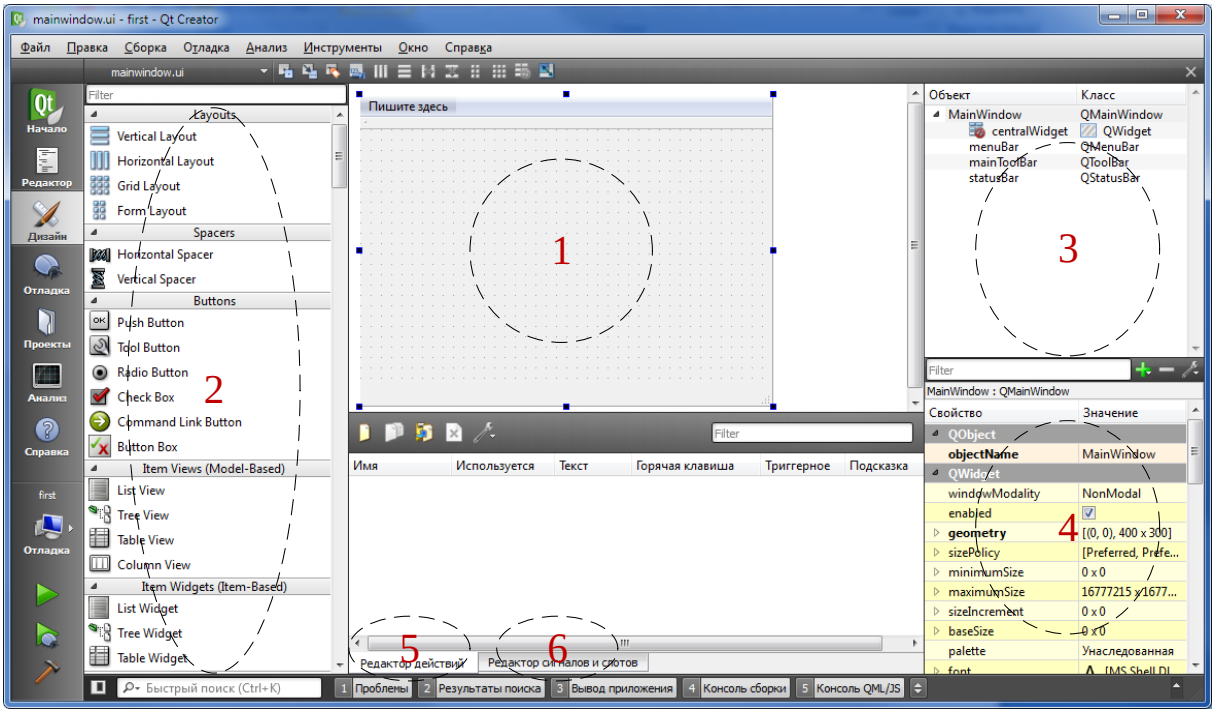

Рис. 6. Разработка интерфейса пользователя

На рис. 6 пунктиром обозначены:

1) форма, расположенная в центре окна;
2) панель виджетов (Wiget Box);
3) инспектор объектов (Object Inspector);
4) редактор свойств (Property Editor);
5) редактор действий;
6) редактор сигналов и слотов.

На заготовке формы уже размещены элементы: Меню (menuBar) и Панель инструментов (mainToolBar). Пусть они остаются без изменений.

Разместите на форме две кнопки (Push Button). Для размещения виджетов на форме следует перетащить их мышкой на форму из панели виджетов.

Отредактируйте надписи на кнопках и их имена. Для этого выберите первую из кнопок мышкой на форме (или в инспекторе объектов). В редакторе свойств найдите свойство *text* и измените его значение с *PushButton* на *Очистить*. Измените имя первой кнопки, для этого свойству *objectName* задайте значение *pushButtonClear*.

Аналогично задайте для второй кнопки: свойству *text* – значение *Копировать*, свойству *objectName* – значение *pushButtonCopy*.

Добавьте еще два виджета Line Edit – Поля редактирования. Для первого поля редактирования задайте имя *lineEdit_1*, свойству *text* задайте значение *источник*. Свойству *text* второго поля редактирования с именем *lineEdit_2* задайте значение *получатель*.

После выполнения всех манипуляций окно программы примет вид, аналогичный приведенному на рис. 7.

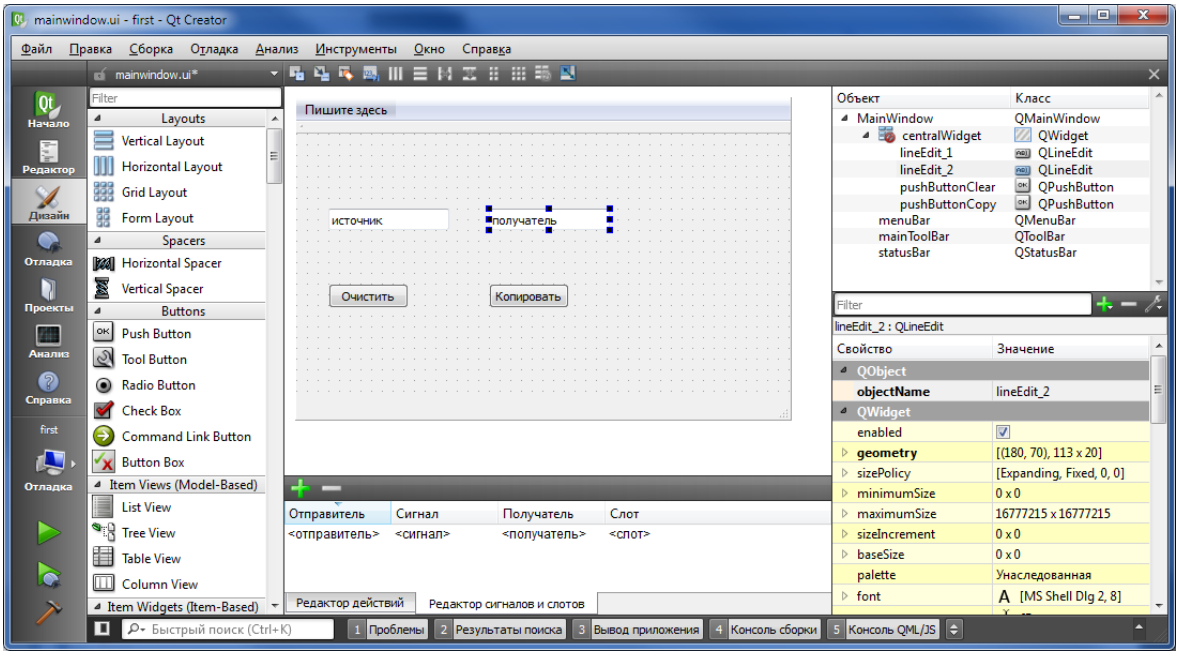

Рис. 7. Разработка интерфейса пользователя

Проверьте правильность выполнения, запустите программу. Перед запуском программы Qt Creator предлагает сохранить изменения в файлах проекта, показывая окно «Сохранение изменений». Сохраните проект, нажав на кнопку «Сохранить все».

Пока программа не выполняет полезных действий, но мы уже познакомились с виджетами, научились изменять их свойства в редакторе свойств и разработали вполне достойный графический интерфейс с кнопками и полями редактирования.

Измените заголовок окна с «MainWindow», например, на «Hello, World!». Выполните это самостоятельно.

3.3. Редактирование свойств виджетов из кода программы

Добавьте на форму элемент Надпись. Для этого из панели виджетов перетащите на форму компонент Label. Переименуйте его. Для этого в редакторе свойств определите свойство *objectName* значением *labelResult*.

Увеличьте элемент Надпись, чтобы в нее влезал выводимый текст. Для этого «схватите» границу надписи мышкой и растяните.

Измените текст надписи из кода программы, открыв файл mainwindow.cpp, затем после строки

```
ui->setupUi(this);
```

добавьте строку

```
ui->labelResult->setText("Результат!");
```

При редактировании кода программы среда разработки Qt Creator выполняет автоматическое дополнение кода, предлагая список соответствующих альтернатив, как показано на рис. 8.

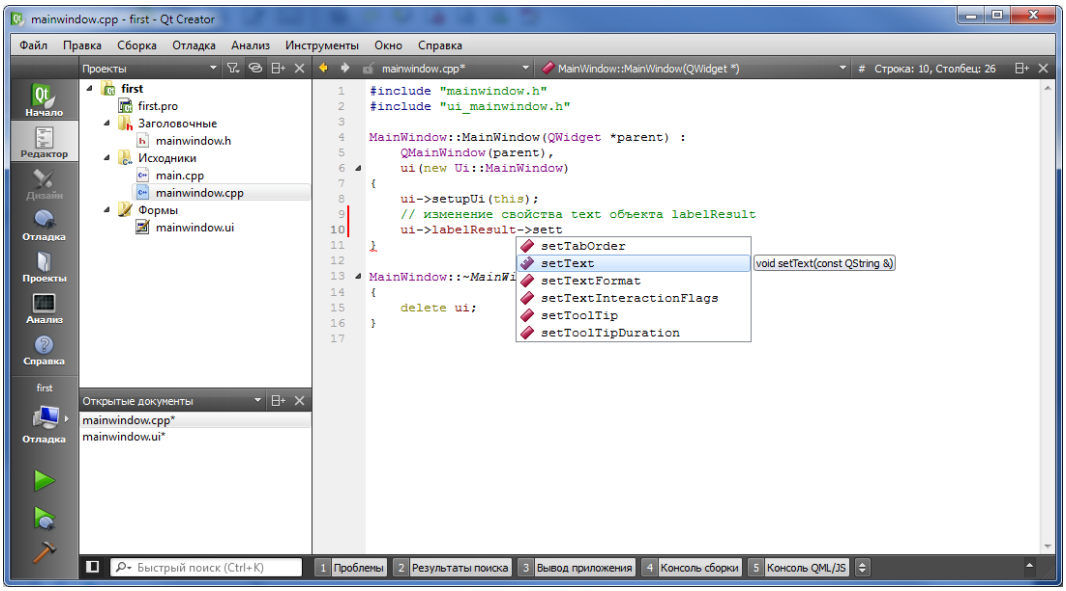

Рис. 8. Автоматическое дополнение кода

Редактор среды разработки выделяет различными цветами синтаксические элементы, что упрощает визуальное восприятие и облегчает обнаружение синтаксических ошибок.

После изменений листинг файла mainwindow.cpp должен иметь следующий вид:

```
1:   #include "mainwindow.h"
2:   #include "ui_mainwindow.h"
3:
4:   MainWindow::MainWindow(QWidget *parent) :
5:       QMainWindow(parent),
6:       ui(new Ui::MainWindow)
7:   {
8:       ui->setupUi(this);
9:       // изменение свойства text объекта labelResult
10:      ui->labelResult->setText("Результат!");
11:  }
12:
13:  MainWindow::~MainWindow()
14:  {
15:      delete ui;
16:  }
```

Пояснения к программе

В первых двух строках (1 и 2) подключаются заголовочные файлы модулей.

Далее следует код конструктора MainWindow. Код, расположенный между фигурными скобками (строки 7–11), будет выполняться при запуске программы.

Строка 9 – однострочный комментарий. Строки, начинающиеся с двух косых черточек (слэшей), служат для пояснений текста программы и игнорируются компилятором.

Строка 10 – вызов метода (функции), изменяющего свойство *text* элемента *labelResult*. В скобках указан аргумент функции – строка «Результат!». Для корректного обращения к виджетам созданной формы перед их именем следует писать `ui->`

Строки 13–16 связаны с уничтожением окна.

Проверьте корректность внесенных дополнений, запустите программу. Текст надписи на форме «*TextLabel*» изменяется на «*Результат!*» из кода программы.

Итак, мы научились изменять размеры компонентов на форме с помощью мыши, обращаться к свойствам элементов из кода программы и изменять их.

3.4. Настройка поведения формы. Механизм сигналов и слотов

Продолжим развивать наше приложение. Сделаем так, чтобы при нажатии на кнопку «Очистить» на форме очищались поля редактирования. Для этого откройте Редактор сигналов и слотов, нажмите на зеленый крестик и добавьте новую запись.

Используя двойной клик мышкой и выпадающие списки, измените значения полей созданной строки. В поле Отправитель установите *pushButtonClear*, в поле Сигнал выберите *clicked()*, в поле Получатель выберите *lineEdit_1*, в поле Слот выберите *clear()*.

Нажмите еще раз на зеленый крестик и добавьте еще одну строку. Настройте ее аналогично, только в поле Получатель выберите *lineEdit_2*. Окно среды разработки примет вид, аналогичный приведенному на рис. 9.

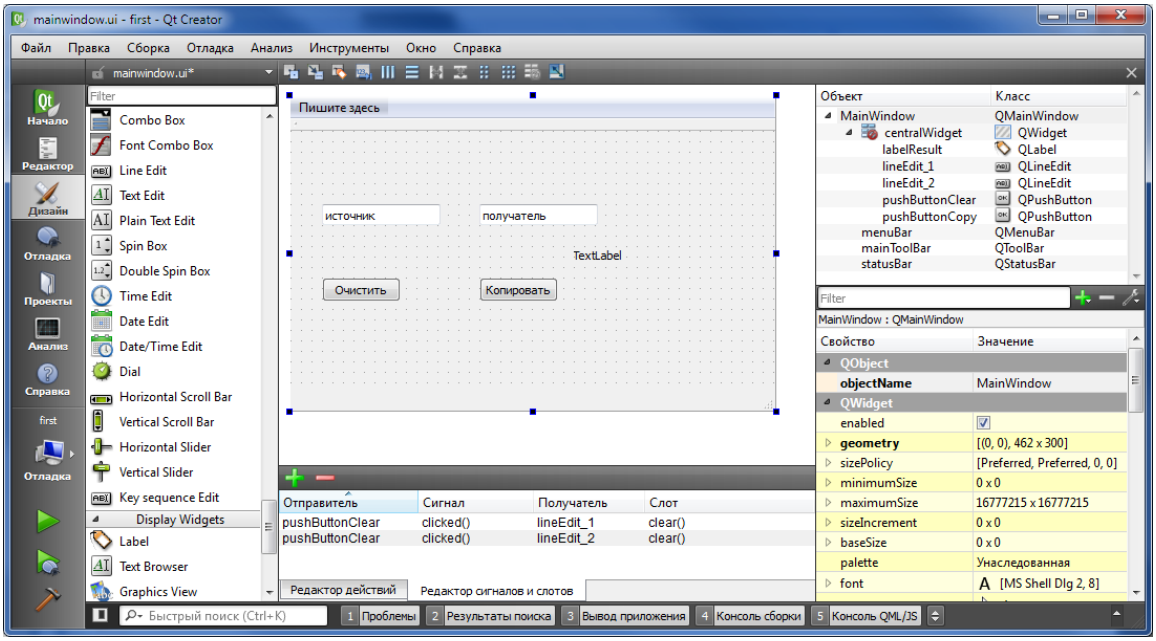

Рис. 9. Добавление сигналов и слотов

Проверьте правильность внесенных дополнений, сохраните проект и запустите программу. При нажатии на кнопку «Очистить» происходит удаление текста из полей редактирования.

Механизм сигналов и слотов позволяет реализовать взаимодействие интерфейса и пользователя. Элементы интерфейса реагируют на действия пользователя, посылая сигналы. В Qt под сигналами подразумеваются методы, которые в состоянии осуществить пересылку сообщений. Сигналы определяются в классе, как обычные методы, только без реализации. С точки зрения программиста они являются только прототипами методов, содержащихся в заголовочном файле определения класса.

Библиотека предоставляет большое число уже готовых сигналов для существующих элементов управления, однако иногда возникает необходимость реализации новых сигналов в своих классах.

Слоты – это методы, которые присоединяются к сигналам. Для простоты можно считать, что слоты являются функциями, способными принимать сигналы. Сигналы не обязательно соединять со слотом. Если соединение не произошло, то сигнал просто не будет обрабатываться. Один сигнал можно соединить с несколькими слотами, один слот, в свою очередь, может быть соединен со многими сигналами [1].

Сигналы и слоты – это концепция, отличающая Qt от C++ Builder и Delphi, которые используют функции обратного вызова для обработки событий формы.

3.5. Создание своего обработчика событий

Создайте прототип своего слота, для этого откройте заголовочный файл mainwindow.h и после строк

```
private:
    Ui::MainWindow *ui;
```

добавьте следующие строки:

```
public slots:
    void MyEventHandler();
```

После изменений листинг файла mainwindow.h должен иметь следующий вид:

```
1:  #ifndef MAINWINDOW_H
2:  #define MAINWINDOW_H
3:
4:  #include <QMainWindow>
5:
6:  namespace Ui {
7:  class MainWindow;
8:  }
9:
10: class MainWindow : public QMainWindow
11: {
12:     Q_OBJECT
13:
14: public:
15:     explicit MainWindow(QWidget *parent = 0);
16:     ~MainWindow();
17:
18: private:
```

```
19:      Ui::MainWindow *ui;
20:
21:  public slots:
22:      void MyEventHandler();
23:  };
24:
25:  #endif // MAINWINDOW_H
```

Создайте тело слота, для этого откройте файл mainwindow.cpp и в его конец допишите следующие строки:

```
// тело слота (т. е. функция обработки сигнала)
void MainWindow::MyEventHandler()
{
    ui->lineEdit_2->setText(ui->lineEdit_1->text());
}
```

Свяжите сигнал со слотом, для этого в конец конструктора MainWindow (файл mainwindow.cpp) добавьте следующие строки:

```
// свяжем сигнал со слотом
QObject::connect(ui->pushButtonCopy,
   SIGNAL(clicked()),this, SLOT(MyEventHandler()));
```

После изменений листинг файла mainwindow.cpp должен иметь следующий вид:

```
1:   #include "mainwindow.h"
2:   #include "ui_mainwindow.h"
3:
4:   MainWindow::MainWindow(QWidget *parent) :
5:       QMainWindow(parent),
6:       ui(new Ui::MainWindow)
7:   {
8:       ui->setupUi(this);
9:       // изменение свойства text объекта labelResult
10:      ui->labelResult->setText("Результат!");
11:      // свяжем сигнал со слотом
12:      QObject::connect(ui->pushButtonCopy,
13:         SIGNAL(clicked()),this, SLOT(MyEventHandler()));
14:  }
15:
16:  MainWindow::~MainWindow()
17:  {
18:      delete ui;
19:  }
20:
21:  // тело слота (т. е. функция обработки сигнала)
22:  void MainWindow::MyEventHandler()
```

```
23:  {
24:      ui->lineEdit_2->setText(ui->lineEdit_1->text());
25:  }
26:
```

Проверьте текст программы, сохраните проект, запустите выполнение программы. Программа должна работать следующим образом: в поле редактирования *источник* пользователь вводит текст. При нажатии на кнопку «Копировать» текст из поля *источник* копируется в поле редактирования *получатель*.

3.6. Создание своего сигнала

Откройте файл mainwindow.h и в класс MainWindow добавьте раздел сигналов и прототип сигнала:

```
signals:
    void MySignal(QString);
```

Эта запись говорит о том, что в классе MainWindow может быть испущен сигнал MySignal с параметром типа QString. У сигналов должен быть прототип, но не должно быть определения, так как сигнал – это не метод класса.

После изменений листинг файла mainwindow.h должен иметь следующий вид:

```
1:   #ifndef MAINWINDOW_H
2:   #define MAINWINDOW_H
3:
4:   #include <QMainWindow>
5:
6:   namespace Ui {
7:   class MainWindow;
8:   }
9:
10:  class MainWindow : public QMainWindow
11:  {
12:      Q_OBJECT
13:
14:  public:
15:      explicit MainWindow(QWidget *parent = 0);
16:      ~MainWindow();
17:
18:  private:
19:      Ui::MainWindow *ui;
20:
```

```
21:  public slots:
22:      void MyEventHandler();
23:
24:  signals:
25:      void MySignal(QString);
26:  };
27:
28:  #endif // MAINWINDOW_H
```

Внесите изменения в файл mainwindow.cpp. Замените строку

```
ui->lineEdit_2->setText(ui->lineEdit_1->text());
```

на строку

```
emit MySignal(ui->lineEdit_1->text());
```

Команда emit испускает сигнал MySignal с содержимым поля lineEdit_1 в качестве параметра.

В конструкторе класса MainWindow вставьте следующие две строки:

```
QObject::connect(this, SIGNAL(MySignal(QString)),
                 ui->lineEdit_2, SLOT(setText(QString)));
QObject::connect(this, SIGNAL(MySignal(QString)),
                 ui->labelResult, SLOT(setText(QString)));
```

Таким образом, мы соединяем наш собственный сигнал MySignal (QString) со слотами сразу двух объектов: поля редактирования lineEdit_2 и надписи labelResult.

После всех изменений листинг файла mainwindow.cpp должен иметь следующий вид:

```
1:   #include "mainwindow.h"
2:   #include "ui_mainwindow.h"
3:
4:   MainWindow::MainWindow(QWidget *parent) :
5:       QMainWindow(parent),
6:       ui(new Ui::MainWindow)
7:   {
8:       ui->setupUi(this);
9:       // изменение свойства text объекта labelResult
10:      ui->labelResult->setText("Результат!");
11:
12:      QObject::connect(ui->pushButtonCopy,
13:         SIGNAL(clicked()),this, SLOT(MyEventHandler()));
14:      QObject::connect(this, SIGNAL(MySignal(QString)),
15:         ui->lineEdit_2, SLOT(setText(QString)));
16:      QObject::connect(this, SIGNAL(MySignal(QString)),
17:         ui->labelResult, SLOT(setText(QString)));
18:  }
19:
```

```
20:  MainWindow::~MainWindow()
21:  {
22:      delete ui;
23:  }
24:
25:  // тело слота (т. е. функции обработки сигнала)
26:  void MainWindow::MyEventHandler()
27:  {
28:      emit MySignal(ui->lineEdit_1->text());
29:  }
30:
```

Сохраните проект и запустите программу. Программа работает следующим образом: пользователь вводит текст в поле *источник* и нажимает на кнопку «Копировать». Содержимое поля *источник* копируется в поле *получатель* и отображается в надписи labelResult.

В программе происходит следующее: при нажатии на кнопку «Копировать» запускается слот void MainWindow::MyEventHandler(), в нем командой emit MySignal(ui->MyLineEdit1->text()) испускается сигнал MySignal(QString) с содержимым поля MyLineEdit1 в качестве параметра. Следующие две строки:

```
QObject::connect(this, SIGNAL(MySignal(QString)),
                 ui->lineEdit_2, SLOT(setText(QString)));
QObject::connect(this, SIGNAL(MySignal(QString)),
            ui->labelResult, SLOT(setText(QString)));
```

соединяют сигнал MySignal(QString) со слотами setText(QString) сразу двух объектов, поля lineEdit_2 и надписи labelResult. При срабатывании сигнала MySignal(QString) содержимое поля lineEdit_1 передастся слотам setText(QString), которые напечатают в поле редактирования и надписи полученный от сигнала текст.

Итак, продемонстрировано создание сигнала и связывание его со слотом.

3.7. Преобразование строк и математические операции

В очередной раз модернизируем учебную программу. Добавьте на форму еще одну кнопку. Отредактируйте свойства кнопки *objectName* и *text*. Присвойте кнопке имя *pushButtonCalc*. Задайте надпись на кнопке *Расчет*.

Измененная программа должна работать следующим образом: пользователь вводит в поля редактирования lineEdit_1 и lineEdit_2 числа. При нажатии на кнопку «Расчет» в labelResult выводится результат: корень из суммы чисел, введенных в поля lineEdit_1 и lineEdit_2.

Откройте файл MainWindow.h и добавьте в раздел public slots: прототип слота:

```
void CalcHandler();
```

Добавьте в раздел signals: сигнал

```
void CalcSignal();
```

Таким образом, листинг файла MainWindow.h примет следующий вид:

```
1:    #ifndef MAINWINDOW_H
2:    #define MAINWINDOW_H
3:
4:    #include <QMainWindow>
5:
6:    namespace Ui {
7:    class MainWindow;
8:    }
9:
10:   class MainWindow : public QMainWindow
11:   {
12:       Q_OBJECT
13:
14:   public:
15:       explicit MainWindow(QWidget *parent = 0);
16:       ~MainWindow();
17:
18:   private:
19:       Ui::MainWindow *ui;
20:
21:   public slots:
22:       void MyEventHandler();
23:       void CalcHandler();
24:
25:   signals:
26:       void MySignal(QString);
27:       void CalcSignal();
28:   };
29:
30:   #endif // MAINWINDOW_H
31:
```

Отредактируйте файл MainWindow.cpp. В конструктор класса добавьте строку, связывающую слот CalcHandler() и сигнал clicked(), испускаемый при нажатии на кнопку pushButtonCalc:

```
QObject::connect(ui->pushButtonCalc,
                 SIGNAL(clicked()),this,SLOT(CalcHandler()));
```

Добавьте слот – функцию обработки, которая будет выполнять необходимые математические операции:

```
void MainWindow::CalcHandler()
{
}
```

Операторы, помещенные между фигурными скобками, будут выполняться при нажатии на кнопку pushButtonCalc. Допишите команды функции CalcHandler() самостоятельно. Возможный вариант реализации данной функции приведен в листинге файла mainwindow.cpp.

В итоге листинг файл mainwindow.cpp должен принять следующий вид:

```
1:  #include "mainwindow.h"
2:  #include "ui_mainwindow.h"
3:
4:      // подключение библиотеки
5:      // математических функций
6:  #include "math.h"
7:
8:
9:  MainWindow::MainWindow(QWidget *parent) :
10:     QMainWindow(parent),
11:     ui(new Ui::MainWindow)
12: {
13:     ui->setupUi(this);
14:     // Изменение свойства text объекта labelResult
15:     ui->labelResult->setText("Результат!");
16:     // Свяжем сигнал со слотом
17:     QObject::connect(ui->pushButtonCopy,
18:        SIGNAL(clicked()),this, SLOT(MyEventHandler()));
19:     QObject::connect(this, SIGNAL(MySignal(QString)),
20:        ui->lineEdit_2, SLOT(setText(QString)));
21:     QObject::connect(this, SIGNAL(MySignal(QString)),
22:        ui->labelResult, SLOT(setText(QString)));
23:     QObject::connect(ui->pushButtonCalc,
24:        SIGNAL(clicked()),this,SLOT(CalcHandler()));
25: }
26:
27: MainWindow::~MainWindow()
28: {
29:     delete ui;
30: }
31:
32: // Тело слота (т.е. функция обработки сигнала)
33: void MainWindow::MyEventHandler()
34: {
35:     emit MySignal(ui->lineEdit_1->text());
36: }
37:
38: // Расчет корня из суммы чисел
```

```
39:  void MainWindow::CalcHandler()
40:  {
41:   // Объявление трех переменных типа float
42:   float A;
43:   float B;
44:   float S;
45:
46:   // Объявление двух переменных логического типа
47:   bool ok1, ok2;
48:
49:   // Объявление переменной типа QString
50:   QString Res;
51:
52:   // Считывание данных введенных в поля редактирования.
53:   // Производим преобразование строки в число
53:   // логическая переменная ok1 и ok2 сигнализирует о
55:   // успешности выполнении преобразования
56:   A=ui->lineEdit_1->text().toFloat(&ok1);
57:   B=ui->lineEdit_2->text().toFloat(&ok2);
58:
59:   // Индикация корректности ввода данных в поле 1
60:   if (ok1)
61:   {
62:    ui->lineEdit_1->setStyleSheet("background-color:white");
63:   }
64:   else
65:   {
66:    ui->lineEdit_1->setStyleSheet("background-color:red");
67:   }
68:
69:   // Индикация корректности ввода данных в поле 2
70:   if (ok2)
71:   {
72:    ui->lineEdit_2->setStyleSheet("background-color:white");
73:   }
74:   else
75:   {
76:    ui->lineEdit_2->setStyleSheet("background-color:red");
77:   }
78:
79:      // Проверка корректности ввода данных и вычисление
80:      if (ok1&&ok2)
81:      {
82:          S=sqrt(A+B);
83:          // Преобразование числа в строку
84:          Res.setNum(S);
85:          // Вывод на экран
86:          ui->labelResult->setText(Res);
87:      }
88:      else
89:      {
```

```
90:             // Вывод на экран
91:             ui->labelResult->setText("Ошибка ввода!");
92:         }
93:
94:
    }
```

Пояснения к программе

В строке 6 выполняется подключение заголовочного файла math.h, в котором описана функция вычисления квадратного корня sqrt(), используемая в программе в строке 82.

В строке 80 проверяется корректность ввода двух чисел, и после этого выполняются вычисления. В противном случае (строка 91) на экран выводится сообщение об ошибке ввода.

Проверьте корректность внесенных дополнений в программу, запустите выполнение программы и убедитесь в правильности ее работы.

При вводе в поля редактирования числовых значений и нажатии кнопки «Расчет» метка labelResult отображает результат вычисления. При вводе некорректных данных (например, текста) соответствующее поле редактирования окрашивается в красный цвет, вместо результата выводится сообщение «Ошибка ввода!». Экран получившейся программы приведен на рис. 10 и 11.

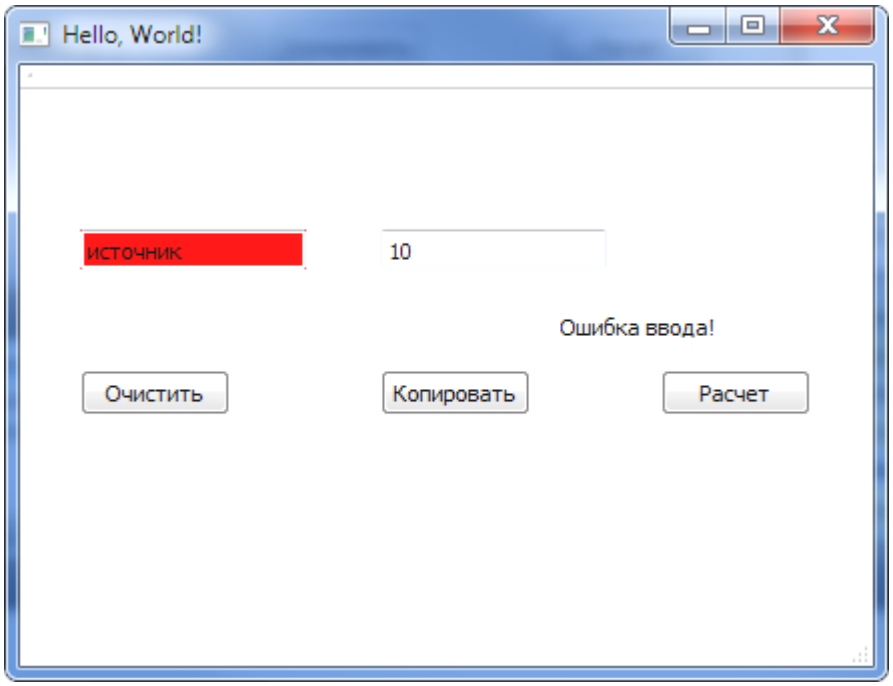

Рис. 10. Экран учебной программы, ошибка ввода

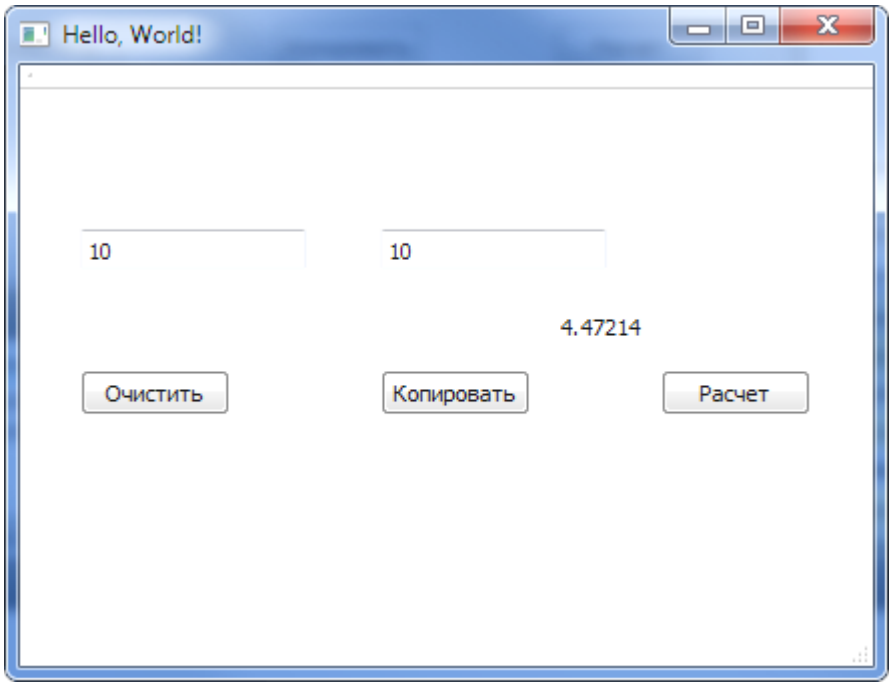

Рис. 11. Экран учебной программы

Итак, на примере учебной программы продемонстрированы основные приемы работы с Qt. Поместите в пояснительную записку экран разработанной учебной программы, при этом в поле редактирования должен быть введен номер варианта.

Далее следует приступать к разработке программы для расчета длины регенерационного участка ВОЛС в соответствии с индивидуальным заданием. Можно последовательно вносить изменения в уже начатый проект учебной программы или создать новый.

4. МЕТОДИКА РАСЧЕТА ДЛИНЫ РЕГЕНЕРАЦИОННОГО УЧАСТКА ВОЛС

Длина регенерационного участка волоконно-оптической линии связи ограничивается двумя явлениями: дисперсий и затуханием [6].

Максимальная длина регенерационного участка по затуханию ($l_з$, км) может быть определена по выражению

$$l_з = \frac{p_{пер} - p_{ф.п.у} - n_{р.с} \cdot a_{р.с} - a_д - M}{\alpha + \frac{a_{н.с}}{l_{с.д}}}, \quad (1)$$

где α – километрическое затухание оптического волокна, дБ/км;

$p_{пер}$ – уровень сигнала на выходе передатчика (регенератора), дБм;

$p_{ф.п.у}$ – уровень сигнала на фотоприемном устройстве, дБм;

$n_{р.с}$ – количество разъемных соединений;

$a_{р.с}$ – средние потери в разъемном соединении, дБ;

$a_д$ – дополнительные потери за счет дисперсии, дБ;

M – системный запас, дБ;

$a_{н.с}$ – средние потери в неразъемном соединении, дБ;

$l_{с.д}$ – строительная длина волоконно-оптического кабеля, км.

Максимальная длина участка регенерации по дисперсии ($l_д$, км) может быть определена по выражению

$$l_д = \frac{2 \cdot \pi \cdot c \cdot \tau_0^2}{\lambda^2 \cdot D_x \cdot \sqrt{1 + 4 \cdot \pi^2 \cdot \Delta\nu^2 \cdot \tau_0^2}}, \quad (2)$$

где c – скорость света в вакууме, $c = 3 \cdot 10^5$ км/с;

τ_0 – ширина оптического импульса на выходе передатчика (зависит от скорости передачи), нс;

λ – длина волны излучения источника, мкм;

$D_х$ – величина хроматической дисперсии, зависящая от типа применяемого волокна, пс/(нм·км);

$\Delta\nu$ – ширина спектра излучения, ГГц.

Характеристики систем передачи приведены в табл. 3.

Таблица 3

Характеристики систем передачи

Параметр	Система передачи			
	STM-1	STM-4	STM-16	STM-64
Скорость передачи, *C*, Мбит/с	155	620	2500	10000
τ_0, нс	1,54	0,386	0,096	0,024

Характеристики оптических волокон приведены в табл. 4

Таблица 4

Характеристики оптических волокон

Параметр оптического волокона в соответствии с рекомендацией МСЭ		Рекомендация МСЭ			
		G.652	G.653	G.655	G.655
Тип волокна		SSF	DSF	–NZDSF	+NZDSF
Окно прозрачности, нм		1300/1550	1500–1600	1530–1565	1530–1565
Коэффициент затухания α, дБ/км	1310 нм	0,34	<1,0	Не нормируется	Не нормируется
	1550 нм	0,22	0,22	0,19–0,22	0,19–0,22
Хроматическая дисперсия D_x, пс(км/нм	1310 нм	±3,5	17–18	Не нормируется	Не нормируется
	1550 нм	17–18	±3,5	–(5–8)	5–8

При тестировании разработанной программы и проведении расчета рекомендуется использовать исходные данные указанных диапазонов:

уровень сигнала на выходе передатчика $p_{пер} = –2…+4$ дБ;

уровень сигнала на фотоприемном устройстве $p_{ф.п.у} = –8…–24$ дБ;

количество разъемных соединений $p_{р.с} = 2$;

средние потери в разъемном соединении $a_{р.с} = 0{,}15…0{,}25$ дБ;

дополнительные потери за счет дисперсии $a_д = 1{,}0$ дБ;

системный запас *M*, обычно принимают 3,0 дБ;

средние потери в неразъемном соединении $p_{н.с} = 0{,}05…0{,}15$ дБ;

строительная длина кабеля $l_{с.д} = 5{,}0…15{,}0$ км;

длина волны излучения данного источника $\lambda = 1{,}310; 1{,}550$ мкм;

ширина спектра излучения $\Delta\nu = 1{,}0…100$ ГГц

Остальные параметры расчета определяются индивидуальным вариантом задания в соответствии с табл. 1–4.

5. ОФОРМЛЕНИЕ АЛГОРИТМА РАБОТЫ ПРОГРАММЫ

Перед выполнением алгоритмов программ рекомендуется ознакомиться с условными обозначениями, приведенными в ГОСТ 19.701–90 «Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения» [7].

При оформлении алгоритма программы рекомендуется использовать редактор Microsoft Visio. Редактор Visio достаточно прост в использовании и содержит богатый набор шаблонов фигур для составления алгоритмов программ [8]. На рис. 12 приведен набор фигур Visio для составления простой блок-схемы.

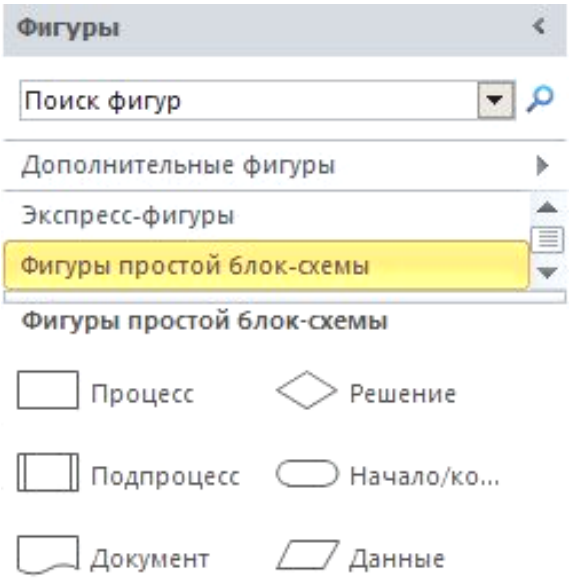

Рис. 12. Фигуры простой блок-схемы редактора Visio

Фигуры Visio представляют собой готовые изображения, которые перетаскиваются из окна «Фигуры» на страницу документа (рис. 13).

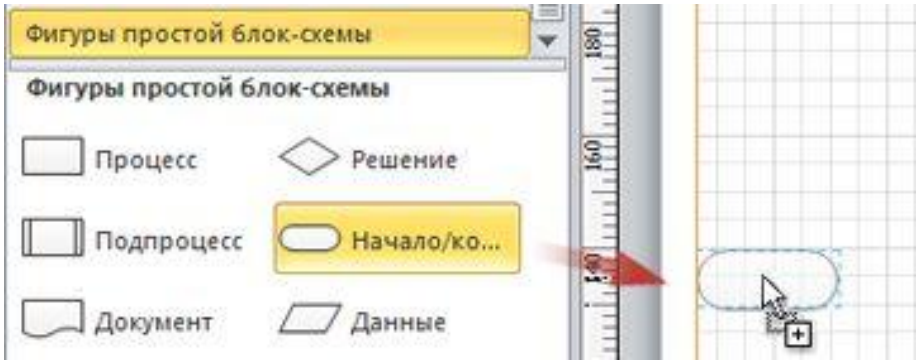

Рис. 13. Добавление элемента в схему

6. РАЗРАБОТКА РУКОВОДСТВА ОПЕРАТОРА

Документ «Руководство оператора» относится к пакету эксплуатационной документации. Основная цель руководства оператора заключается в обеспечении пользователя необходимой информацией для самостоятельной работы с программой или автоматизированной системой.

В ходе выполнения курсовой работы предлагается включить в состав пояснительной записки «Руководство оператора» для разработанной программы по расчету длины регенерационного участка ВОЛС. Перед составлением Руководства рекомендуется ознакомиться с ГОСТ 19.505–79 «Руководство оператора. Требования к содержанию и оформлению» [9].

Руководство оператора должно содержать следующие разделы:

– назначение программы;

– условия выполнения программы;

– выполнение программы;

– сообщения оператору.

В зависимости от особенностей документа допускается объединять отдельные разделы или вводить новые.

Заключение

На завершающем этапе выполнения курсовой работы следует проверить соответствие работы варианту индивидуального задания, соблюдение требований к содержанию и оформлению пояснительной записки. В частности, рекомендуется проверить, что в пояснительной записке:

1) указан номер варианта;

2) приведен экран разработанной учебной программы;

3) содержится алгоритм работы программы для расчета длины регенерационного участка ВОЛС;

4) на форме разработанной программы для расчета длины регенерационного участка ВОЛС указаны: фамилия, имя и номер группы разработчика программы;

5) приведен экран (PrintScreen) разработанной программы, при этом в поля редактирования введены исходные данные, и проведен расчет длины участка регенерации по дисперсии и затуханию;

6) содержится руководство оператора [9];

7) страницы пронумерованы;

Пояснительная записка должна быть сброшюрована.

Список использованных источников

1. *Шлее, М.* Qt 4.8. Профессиональное программирование на C++ / М. Шлее. – СПб. : БХВ – Петербург, 2013.

2. Qt – руководство для новичков / CyberForum.ru [Электронный ресурс]. – URL: http://www.cyberforum.ru/qt/thread79698.html (дата обращения: 30.12.13).

3. CrossPlarform.RU Все о кроссплатформенном программировании [Электронный ресурс]. – URL: http://doc.crossplatform.ru/qt/ (дата обращения: 30.12.13).

4. Qt Documentation [Электронный ресурс]. – URL: http://doc.qt.io.

5. ГОСТ 7.32–2001. Отчет о научно-исследовательской работе. Структура и правила оформления.

6. *Глаголев, С. Ф.* Определение длины регенерационного участка ВОЛС / С. Ф. Глаголев, Л. Н. Кочановский // IV всероссийск. конф. «Современные технологии проектирования, строительства и эксплуатации линейно-кабельных сооружений – СТЛКС» (Санкт-Петербург. Март. 2005 г.) : сборник трудов. – СПб., 2005.

7. ГОСТ 19.701–90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

8. Visio 2007: руководство для начинающих / Microsoft [Электронный ресурс]. – URL: https://support.office.com/ru-ru/article/Visio-2007-руководство-для-начинающих-8deaedcf-cef1-4d2a-8151-008f0a4b4044 (дата обращения: 10.08.16).

9. ГОСТ 19.505–79. Руководство оператора. Требования к содержанию и оформлению.

Дюбов Андрей Сергеевич

КОМПЬЮТЕРНОЕ ОБЕСПЕЧЕНИЕ РАСЧЕТНО-ПРОЕКТНОЙ И ЭКСПЕРИМЕНТАЛЬНО-ИССЛЕДОВАТЕЛЬСКОЙ ДЕЯТЕЛЬНОСТИ

Учебно-методическое пособие по выполнению курсовой работы

Редактор *И. И. Щенсняк*

Компьютерная верстка *Н. А. Ефремовой*

План издания 2017 г., п. 62

Подписано к печати 26.04.2017
Объем 2,0 усл.-печ. л. Тираж 10 экз. Заказ 779

Редакционно-издательский отдел СПбГУТ
191186 СПб., наб. р. Мойки, 61

Отпечатано в СПбГУТ