

**Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное  
учреждение высшего профессионального образования  
«Севастопольский государственный университет»**

**ИССЛЕДОВАНИЕ СПОСОБОВ ПОСТРОЕНИЯ  
ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ НА БАЗЕ QTWIDGETS**

**Методические указания**

к лабораторной работе по дисциплине

**«Кроссплатформенное программирование»**

для студентов, обучающихся по направлению

**09.03.02 “Информационные системы и технологии”**

очной и заочной форм обучения

**Севастополь  
2018**

УДК 004.415.2

**Исследование способов построения интерфейса пользователя на базе QtWidgets.** Методические указания/Сост. Строганов В.А. – Севастополь: Изд-во СевГУ, 2018.–15 с.

Методические указания предназначены для оказания помощи студентам при выполнении лабораторных работ по дисциплине «Кроссплатформенное программирование».

Методические указания составлены в соответствии с требованиями программы дисциплины «Кроссплатформенное программирование» для студентов направления 09.03.02 и утверждены на заседании кафедры «Информационные системы»,  
протокол № от « » \_\_\_\_\_ 2018 г.

**Содержание**

1. Цель работы	4
2. Основные теоретические положения	4
2.1. Пример создания пользовательского графического интерфейса	4
3. Порядок выполнения лабораторной работы и варианты заданий	13
4. Содержание отчета	14
5. Контрольные вопросы	15
Библиографический список	15

## 1. ЦЕЛЬ РАБОТЫ

Изучит основные методики создания графического пользовательского интерфейса с использованием виджетов QtWidgets. Приобрести навыки разработки интерфейса пользователя для приложений на основе фреймворка Qt.

## 2. ОСНОВНЫЕ ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Важной составляющей фреймворка Qt является объемная библиотека стандартных элементов пользовательского интерфейса (виджетов), а также возможность создания собственных виджетов.

Библиотека содержит большинство, наиболее часто используемых при проектировании графического интерфейса виджетов, таких как поля ввода, метки, кнопки, и прочие.

При проектировании графического интерфейса нельзя полагаться на одно разрешение экрана «жестко» устанавливать виджеты на форме. Qt предлагает такие механизмы, как схемы размещения и растяжки, которые используются для гибкого размещения элементов пользовательского интерфейса.

Qt виджеты – базовые элементы пользовательского интерфейса. Каждый виджет наследуется от класса QWidget. Виджет может состоять из множества других виджетов, иметь одну схему размещения и один родительский виджет. Если у виджета нет родителя, то он называется окном. Qt виджеты позволяют создавать графический интерфейс, который будет одинаково функционировать на компьютерах под управлением разных операционных систем и при этом обладать «нативным» интерфейсом.

Схема размещения – гибкий способ размещения виджетов на форме. Схема размещения может содержать множество других схем размещения и виджетов.

Существует 4 основные схемы размещения:

- вертикальная;
- горизонтальная;
- сеточная;
- схема формы.

Типовая архитектура интерфейса Qt приложения подразумевает наличие окна, с схемой размещения, которая содержит множество других схем размещения (различных видов) и виджетов.

### 2.1. Пример создания пользовательского графического интерфейса

Создадим проект простой регистрационной формы.

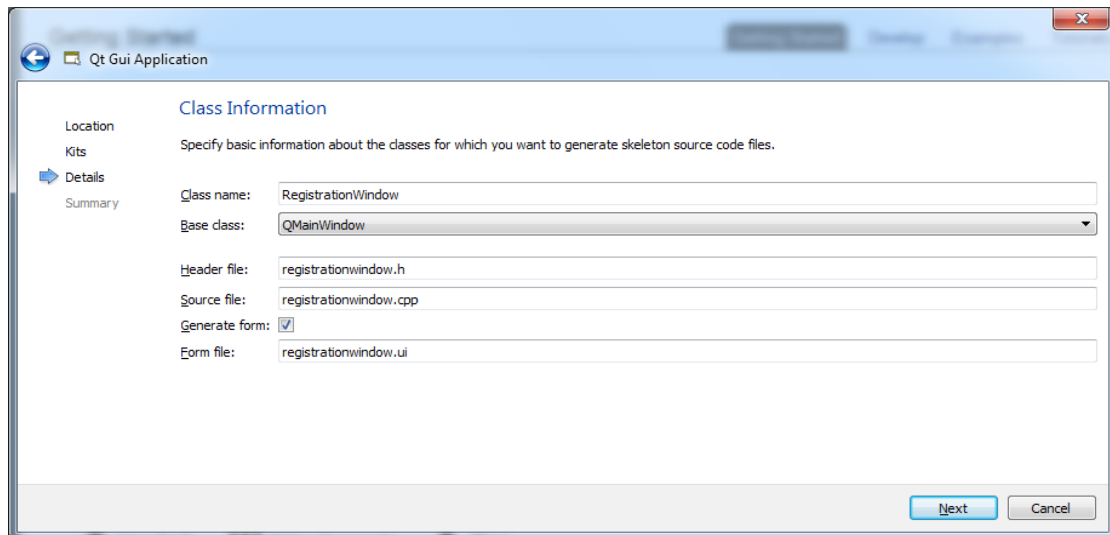


Рисунок 2.1 – Создание проекта

Разместим основные элементы пользовательского интерфейса на форму.

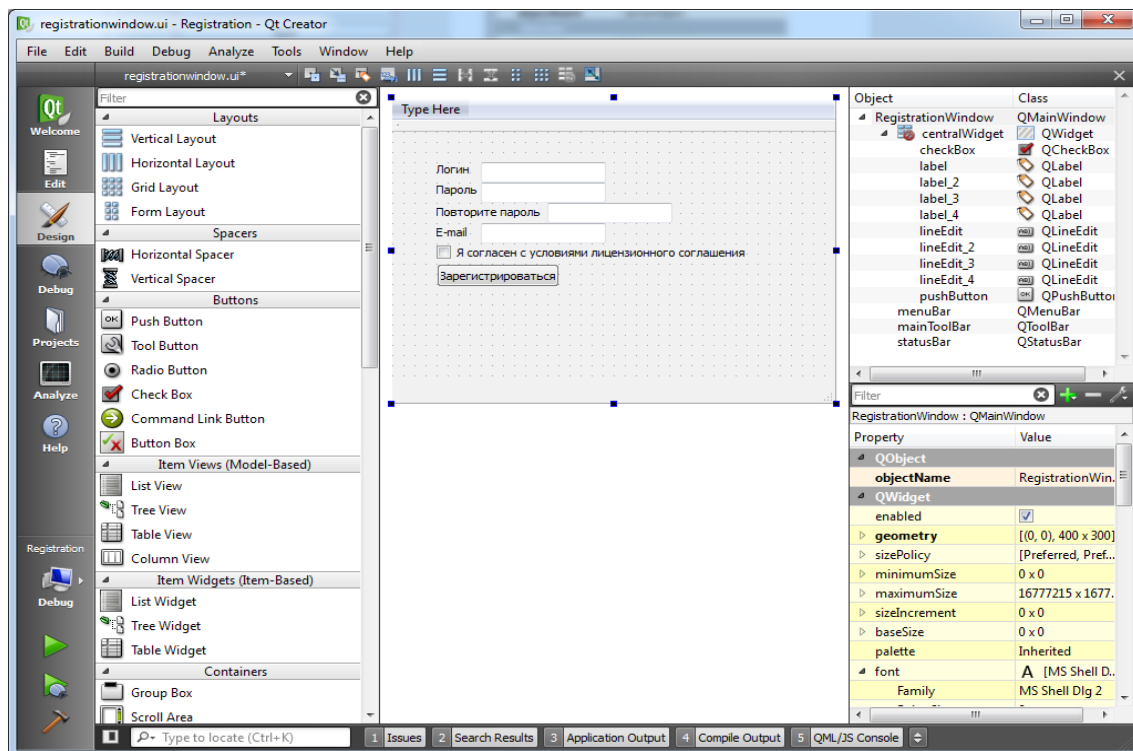


Рисунок 2.2 – Форма с размещенными на ней элементами интерфейса

Такой интерфейс неудачно выглядит, сложен для восприятия и при растягивании окна будет выглядеть еще хуже. Установим для главного окна схему размещения **Form Layout**.

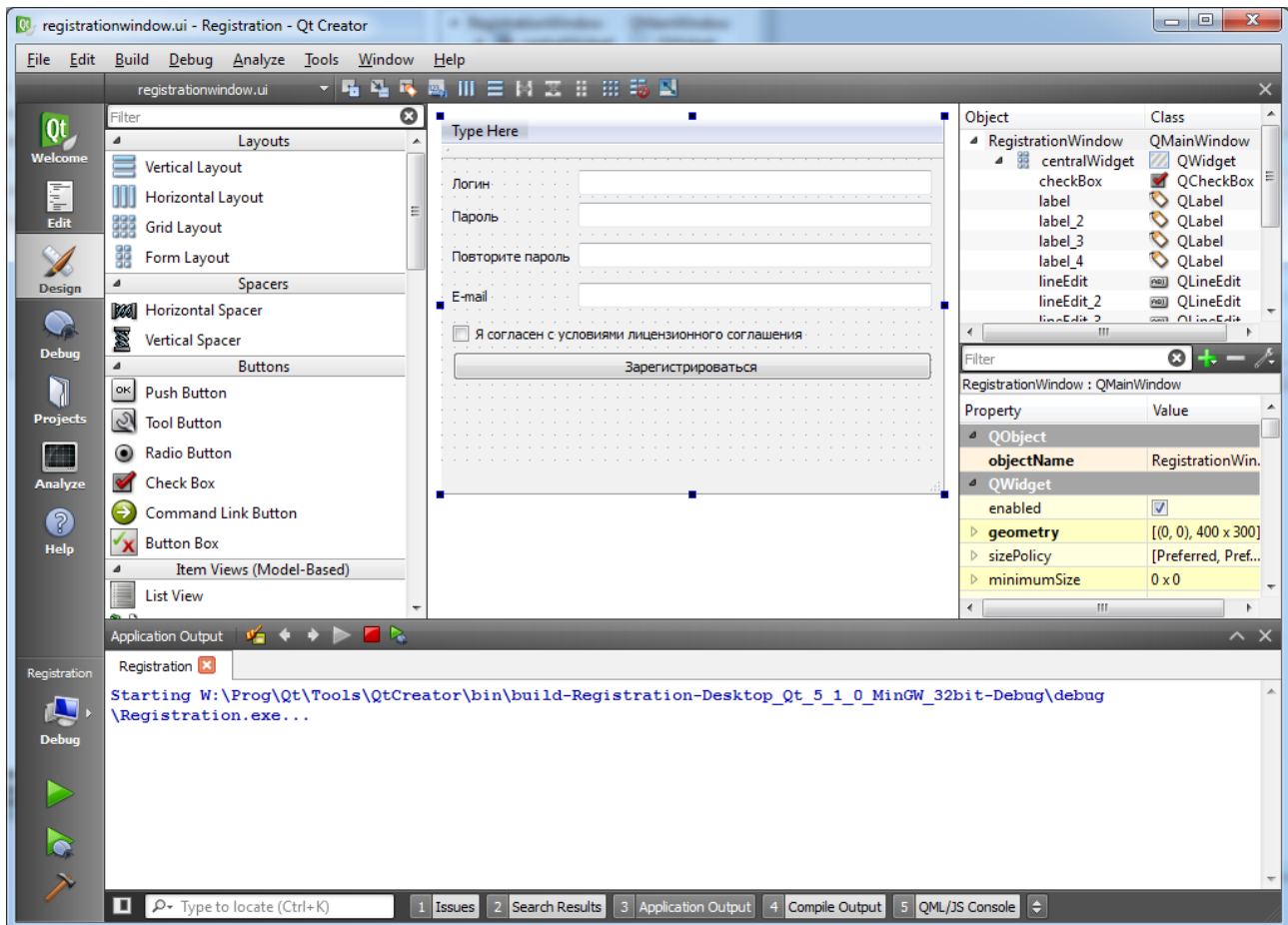


Рисунок 2.3 – Форма с примененной схемой размещения Form Layout

Запускаем наше приложение и видим, что данный интерфейс гораздо более понятен в использовании и изящнее выглядит, однако он тоже не лишен недостатков. Как видно на рисунке 2.4, при вертикальном растягивании остается слишком много пустого места.

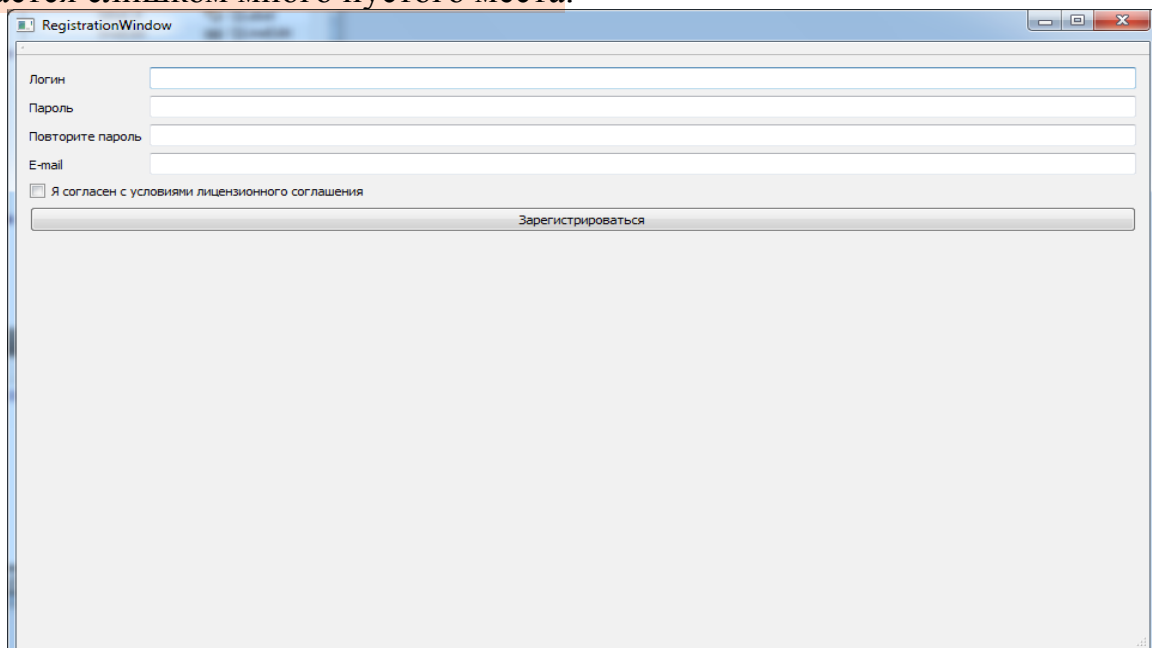


Рисунок 2.4 – Запущенное приложение с примененной схемой размещения Form Layout

Для того, чтобы гибко растянуть элементы по вертикали, добавим на форму элементы пользовательского интерфейса **Vertical Spacer**.

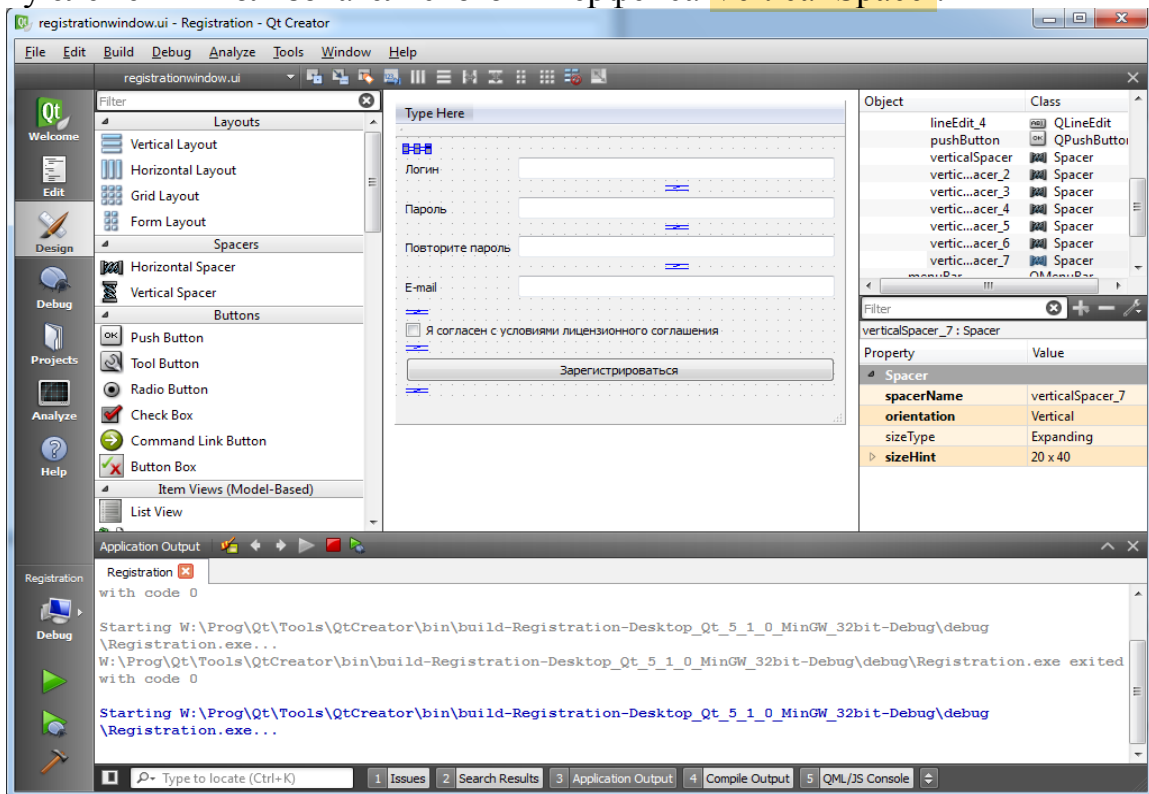


Рисунок 2.5 – Форма с добавленными элементами Vertical Spacer  
При запуске приложения видим, что предыдущий недостаток был устранен.

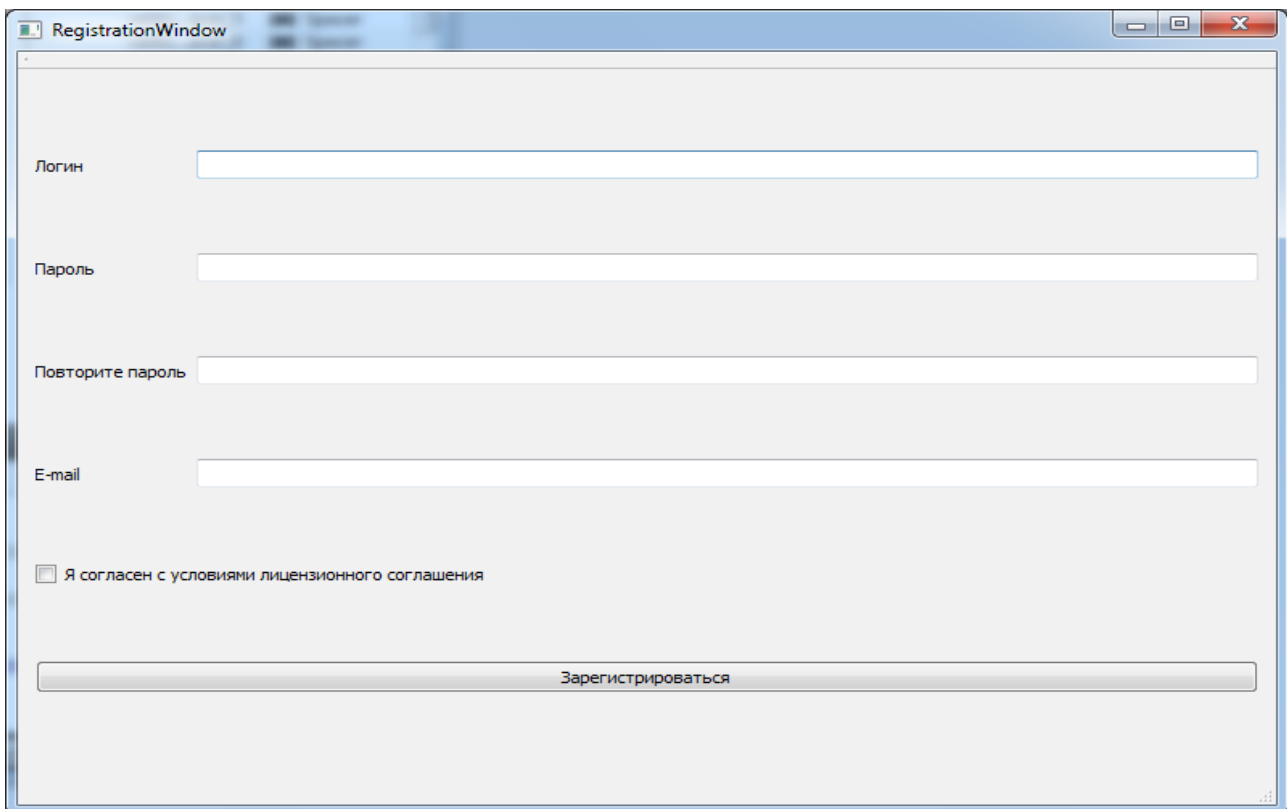


Рисунок 2.6 – Запущенное приложение с добавленными элементами Vertical Spacer

Поскольку нам не нужны такие длинные поля ввода и они смотрятся не очень удачно, более правильным решением будет разместить эти поля в виде таблицы, но схема размещения Form Layout не позволяет разместить элементы в виде таблицы. В этом случае нам придется использовать вложенные схемы размещения.

Уберем схему размещения Form Layout и добавим на форму четыре вертикальных схемы размещения для виджетов.

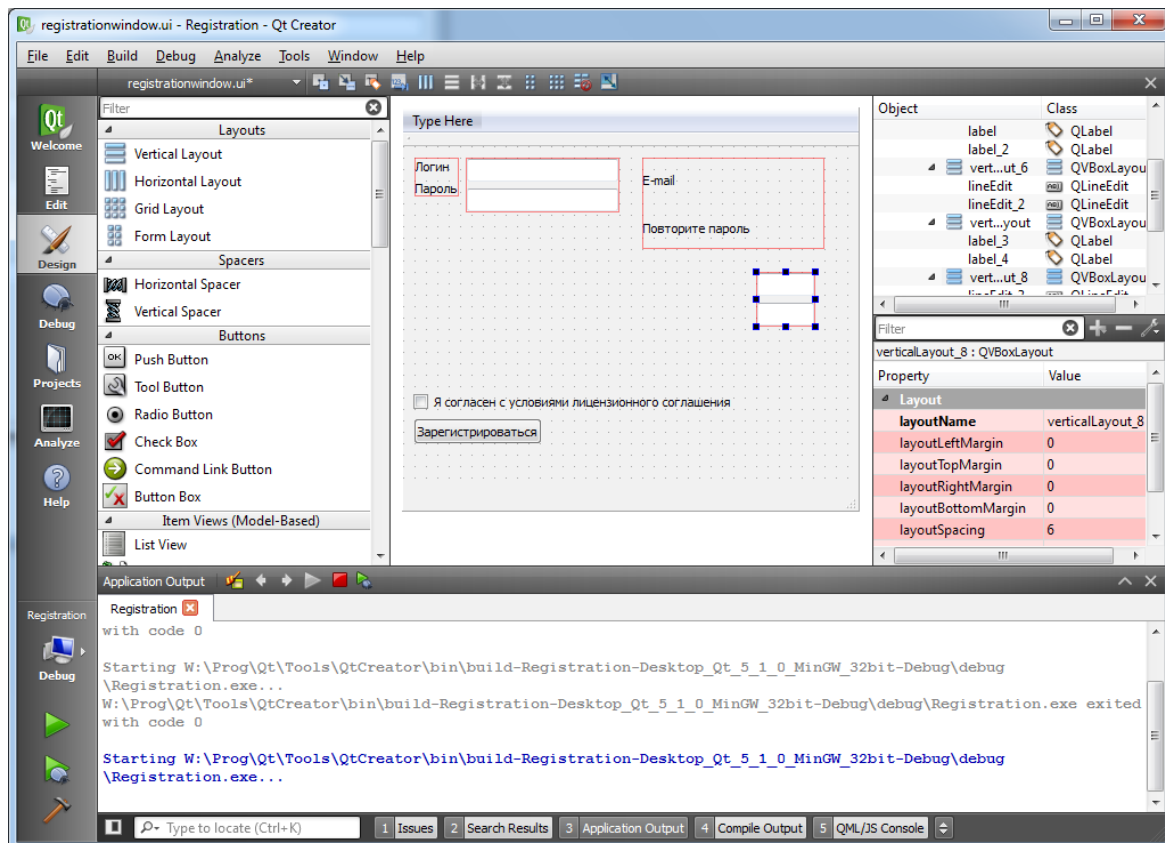


Рисунок 2.7 – Форма с четырьмя вертикальными схемами размещения

Добавим также горизонтальную схему размещения и перетащим на неё четыре ранее созданных.



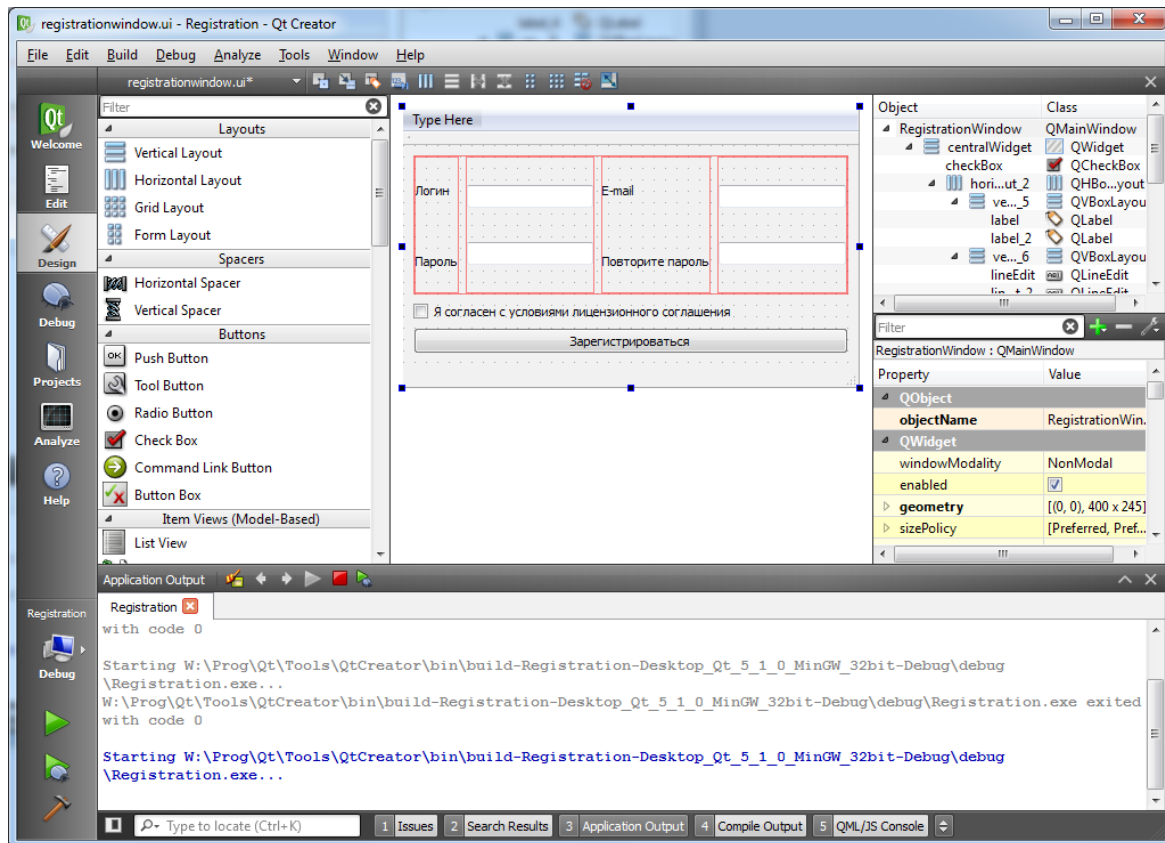


Рисунок 2.8 – Форма с вложенными схемами размещения

Установим для основной формы вертикальную схему размещения и добавим растяжки.

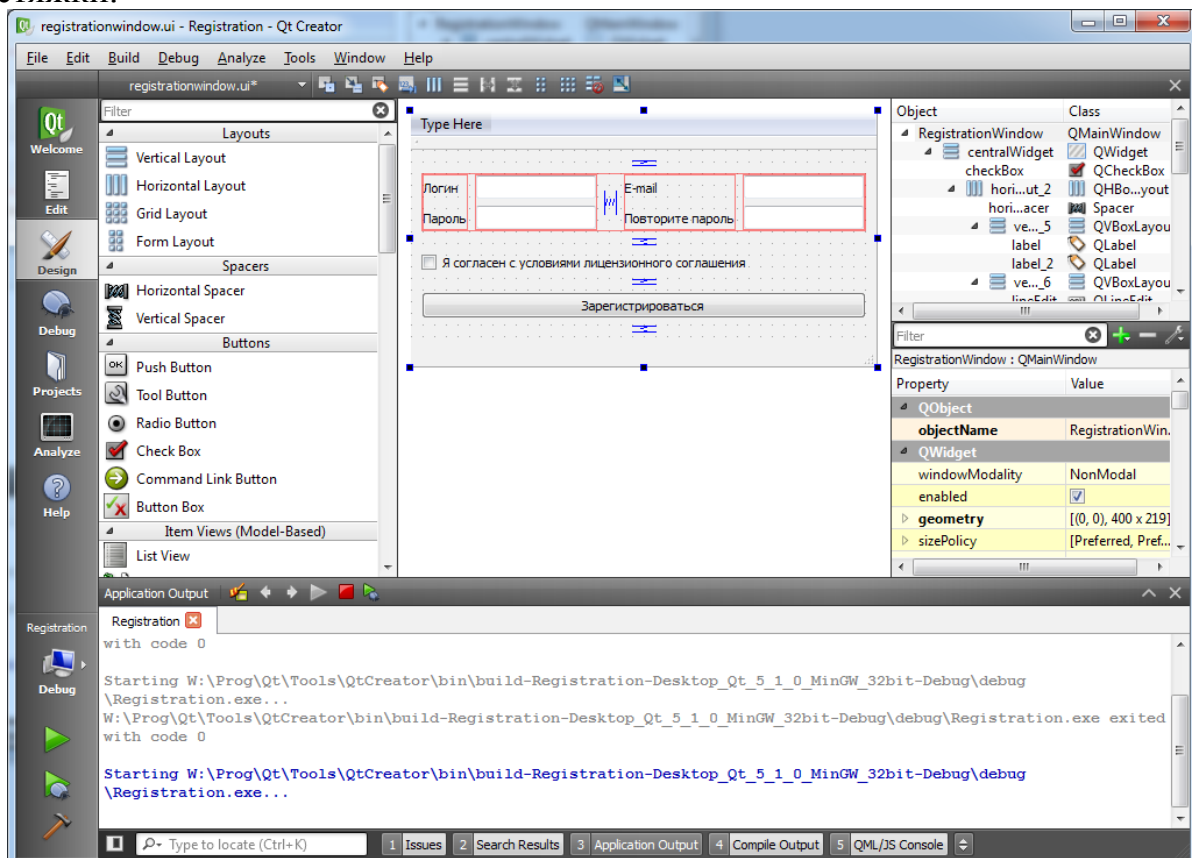


Рисунок 2.9 – Форма с вложенными схемами размещения и растяжками

При запуске приложения получим еще более удобный в использовании интерфейс.

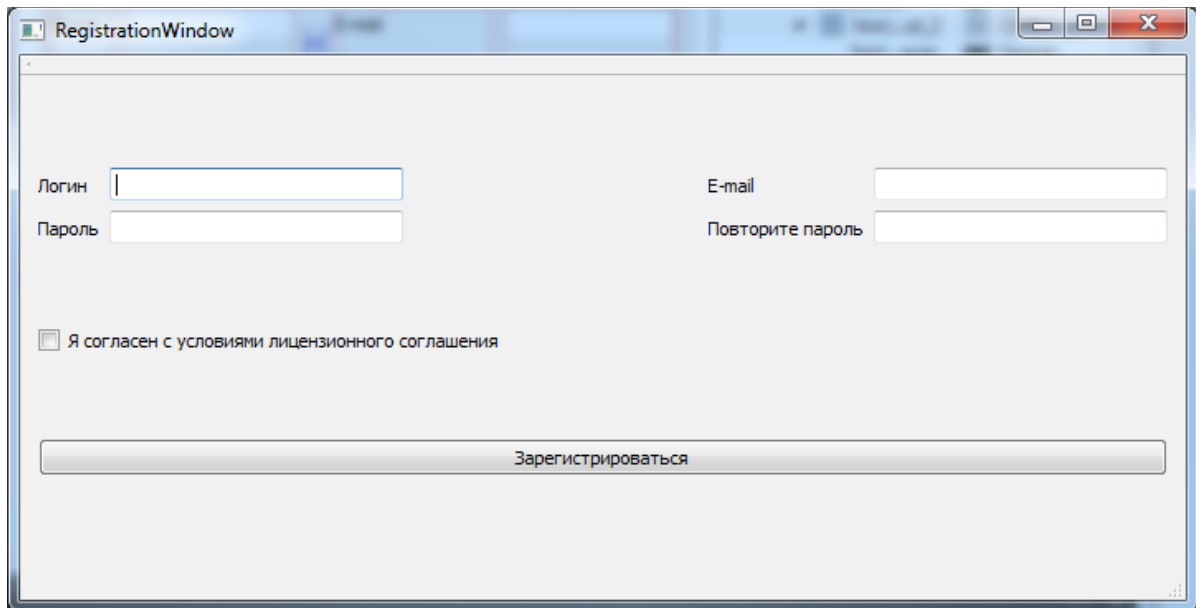


Рисунок 2.10 – Приложение с вложенными схемами размещения

Ранее мы создали форму, используя Qt Designer, однако иногда удобнее работать с кодом напрямую. Создадим такую же форму, без использования Qt Designer.

В первую очередь закомментируем строки кода, в которых устанавливается графический интерфейс файла ui.

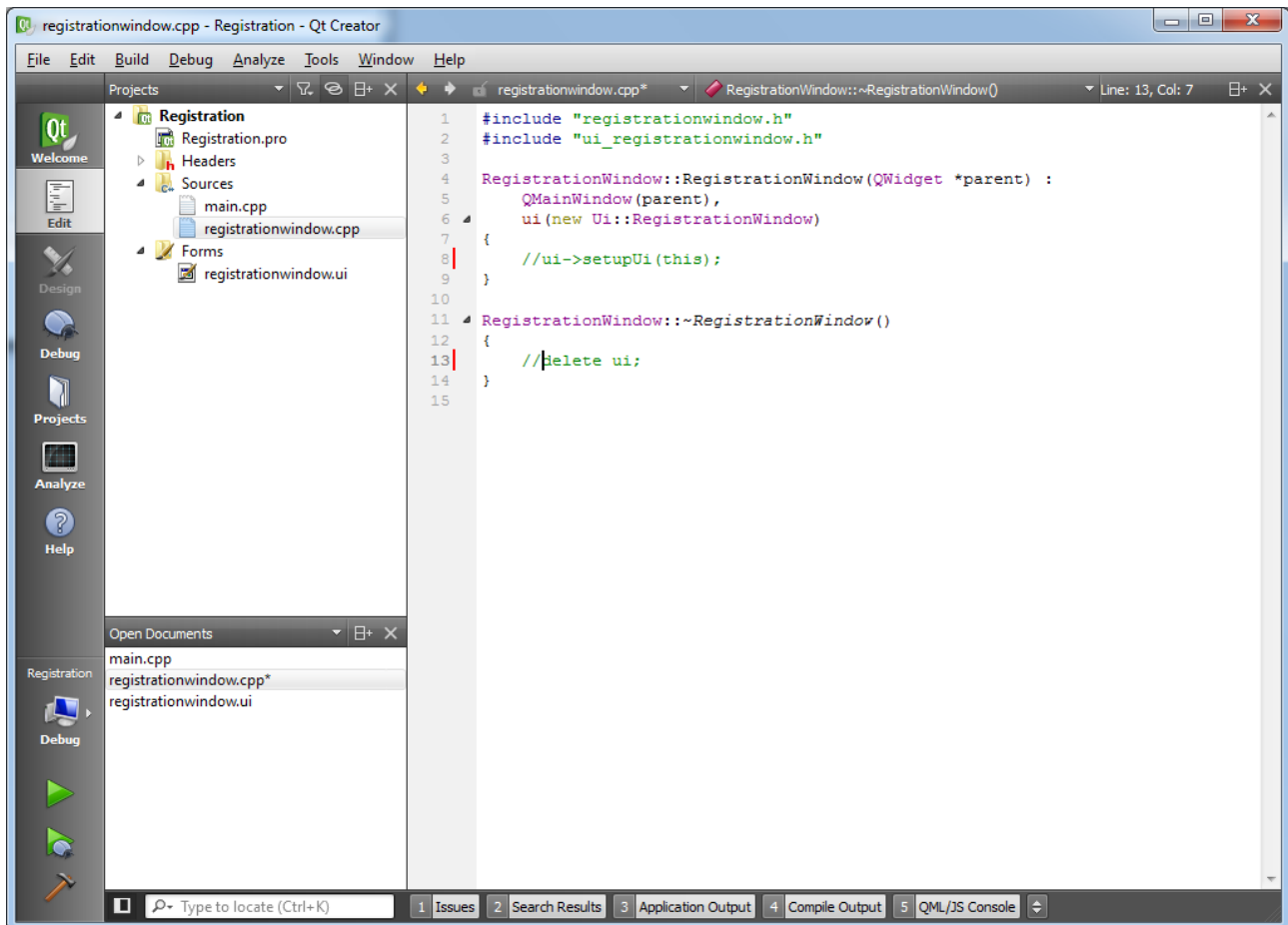


Рисунок 2.11 – Файл registrationwindow.cpp с закомментированными строками инициализации графического интерфейса дизайнера.

Затем подключаем необходимые нам библиотеки и пишем код для создания виджетов и схем размещения. Файл registrationwindow.cpp после этого примет вид:

```
#include "registrationwindow.h"
#include "ui_registrationwindow.h"
```

```
#include "QLabel"
#include "QLineEdit"
#include "QLayout"
#include "QCheckBox"
#include "QPushButton"
```

```
RegistrationWindow::RegistrationWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::RegistrationWindow)
{
    //ui->setupUi(this);

    // Создаем метки для полей "Логин" и "Е-mail"
    QLabel *loginLabel = new QLabel("Логин");
    QLabel *emailLabel = new QLabel("Е-mail");
```

```

// Создаем вертикальную схему размещения для этих меток
// и добавляем их на неё
QLayout *loginAndEmailLabelsLayout = new QVBoxLayout();
loginAndEmailLabelsLayout->addWidget(loginLabel);
loginAndEmailLabelsLayout->addWidget(emailLabel);

// Создаем поля ввода "Логин" и "E-mail"
QLineEdit *loginEdit = new QLineEdit();
QLineEdit *emailEdit = new QLineEdit();

// Создаем вертикальную схему размещения для этих полей ввода
// и добавляем их на неё
QLayout *loginAndEmailEditsLayout = new QVBoxLayout();
loginAndEmailEditsLayout->addWidget(loginEdit);
loginAndEmailEditsLayout->addWidget(emailEdit);

// Аналогичный код для виджетов пароля
QLabel *passwordLabel = new QLabel("Пароль");
QLabel *repeatPasswordLabel = new QLabel("Повторите пароль");

QLayout *passwordLabelsLayout = new QVBoxLayout();
passwordLabelsLayout->addWidget(passwordLabel);
passwordLabelsLayout->addWidget(repeatPasswordLabel);

QLineEdit *passwordEdit = new QLineEdit();
QLineEdit *repeatPasswordEdit = new QLineEdit();

QLayout *passwordEditsLayout = new QVBoxLayout();
passwordEditsLayout->addWidget(passwordEdit);
passwordEditsLayout->addWidget(repeatPasswordEdit);

// Создаем горизонтальную схему размещения для ранее созданных
// вертикальных схем
QHBoxLayout *labelsAndEditsLayout = new QHBoxLayout();
labelsAndEditsLayout->addLayout(loginAndEmailLabelsLayout);
labelsAndEditsLayout->addLayout(loginAndEmailEditsLayout);
labelsAndEditsLayout->addLayout(passwordLabelsLayout);
labelsAndEditsLayout->addLayout(passwordEditsLayout);

// Создаем оставшиеся элементы пользовательского интерфейса
QCheckBox *confirmationCheckBox = new QCheckBox("Я согласен с
условиями пользовательского соглашения.");
QPushButton *registrationButton = new
QPushButton("Зарегистрироваться");

// Создаем основную схему размещения и добавляем на неё все
элементы
QVBoxLayout *centralLayout = new QVBoxLayout();
centralLayout->addLayout(labelsAndEditsLayout);
centralLayout->addWidget(confirmationCheckBox);
centralLayout->addWidget(registrationButton);

// Создаем "центральный" виджет для нашего приложения, и

```

```

устанавливаем его для основного окна (this)
    QWidget *centralWidget = new QWidget();
    centralWidget->setLayout(centralLayout);
    this->setCentralWidget(centralWidget);
}

RegistrationWindow::~RegistrationWindow()
{
    //delete ui;
}

```

После запуска можно убедиться, что форма выглядит так же, как и после использования дизайнера.

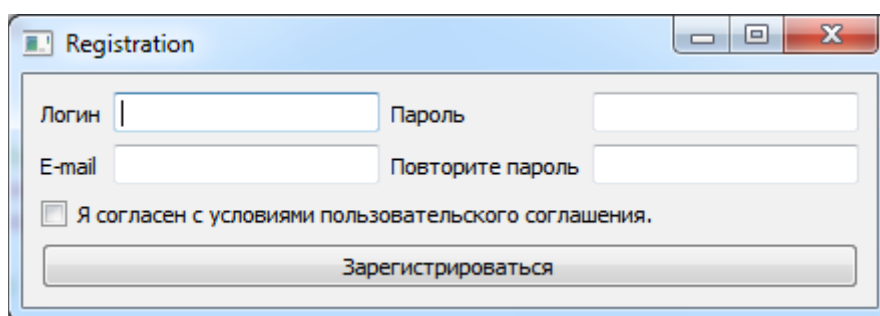


Рисунок 2.12 – Запущенное приложение с графическим интерфейсом описанным в коде программы.

### 3. ПОРЯДОК ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ И ВАРИАНТЫ ЗАДАНИЙ

3.1. Создать проект Qt Gui Application.

3.2. Разработать графический интерфейс приложения согласно варианту задания.

3.3. Используя механизм сигналов и слотов добавить функционал, согласно варианту задания.

3.4. Оформить отчет по работе.

Таблица 3.1 – Варианты заданий

№ варианта	Тип формы	Функционал	Метод разработки
1	1		1 Qt designer
2	2		2 Программно
3	3		3 Qt designer
4	4		4 Программно
5	5		1 Qt designer
6	1		2 Программно
7	2		3 Qt designer
8	3		4 Программно
9	4		1 Qt designer

10	5	2 Программно
11	1	3 Qt designer
12	2	4 Программно
13	3	1 Qt designer
14	4	2 Программно
15	5	3 Qt designer
16	1	4 Программно
17	2	1 Qt designer
18	3	2 Программно
19	4	3 Qt designer
20	5	4 Программно
21	1	2 Qt designer
22	2	3 Программно
23	3	4 Qt designer
24	4	1 Программно
25	5	2 Qt designer
26	1	3 Программно
27	2	4 Qt designer
28	3	1 Программно
29	4	2 Qt designer
30	5	3 Программно

Типы форм:

1) Контактная форма (e-mail (Line Edit), имя (Line Edit), тема (Line Edit), текст сообщения (Text Edit));

2) Форма добавления товара (название (Line Edit), количество (Spin Box), описание (Text Edit), цена (Double Spin Box));

3) Форма отзыва (имя (Line Edit), e-mail (Line Edit), оценка (Radio Button), текст отзыва (Text Edit));

4) Форма добавления новости (заголовок (Line Edit), текст новости (Text Edit), дата (Date Edit));

5) Форма тестирования знаний (3 вопроса, содержание название (Line Edit) и варианты ответа (Check Box, Radio Button, Combo Box)).

Функциональность:

1) Валидация (Наличие текста в полях, соответствие типам вводимых данных);

2) Сохранение введенной информации в файл;

3) Загрузка данных в форму из внешнего файла;

4) Добавление на форму таблицы и добавление в неё введенных данных.

## 4. СОДЕРЖАНИЕ ОТЧЕТА

### 4.1. Цель работы.

- 4.2. Постановка задачи.
- 4.3. Описание использованных элементов пользовательского интерфейса (назначение, основные свойства и методы для работы с ними).
- 4.4. Текст программы.
- 4.5. Выводы по результатам работы.

## **5. КОНТРОЛЬНЫЕ ВОПРОСЫ**

- 5.1. Что такое виджет?
- 5.2. Что такое схема размещения, для чего она нужна?
- 5.3. Какие есть виды схем размещения, в чем их различие?
- 5.4. Сколько виджетов может содержать другой виджет? Сколько схем размещения?
- 5.5. Что такое окно?
- 5.6. Как программно добавить на схему размещения виджет? Как добавить другую схему размещения?

## **БИБЛИОГРАФИЧЕСКИЙ СПИСОК**

- 1. Ж. Бланшет. Qt 3: программирование GUI на C++ / Бланшет Ж., Саммерфилд М. — М. : КУДИЦ — ОБРАЗ, 2005. - 448 с.
- 2. Е.Р. Алексеев. Программирование на языке C++ в среде Qt Creator /Е. Р. Алексеев, Г. Г. Злобин, Д. А. Костюк, О. В. Чеснокова, А. С. Чмыхало.— М.:Альт Линукс, 2015. — 448 с.
- 3. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений на C++/Г. Буч. — М. : БИНОМ ; СПб. : Невский диалект, 2001. - 560 с.
- 4. Шилдт, Г. C++: базовый курс, 3-е издание /Г. Шилдт. — М.: «Вильямс», 2012. — 624 с.
- 5. Шилдт, Г. Полный справочник по C++, 4-е издание /Г. Шилдт. — М.: «Вильямс», 2011. — 800 с.

