Лабораторная работа №1

Создание приложения командной строки в интегрированной среде разработки QtCreator.

Цель работы

Познакомиться с интерфейсом пользователя и основными функциями IDE QtCreator на примере проекта консольного приложения на языке C++ для вывода списка простых чисел.

Задание 1. При необходимости (если работа выполняется не на личном ПК) создать и активировать новую персональную сессию, в качестве названия указав свою фамилию.

Сессия представляет собой именованный набор определённой части настроек среды разработки, таких как: список открытых проектов, открытых окон редактирования, точек останова и наблюдаемых выражений отладчика и т.п. Настройки редакторов, комплекты сборок, конфигурации проектов не являются частью данных сессии и не меняются при их переключении.

Список доступных сессий отображается в панели Начало/Проекты окна среды разработки (рис. 1). Переключить сессию можно, выбрав соответствующий пункт в списке. Правый щелчок на элементе списка открывает локальное меню, позволяющее скопировать, переименовать, или удалить сеанс (рис. 2).

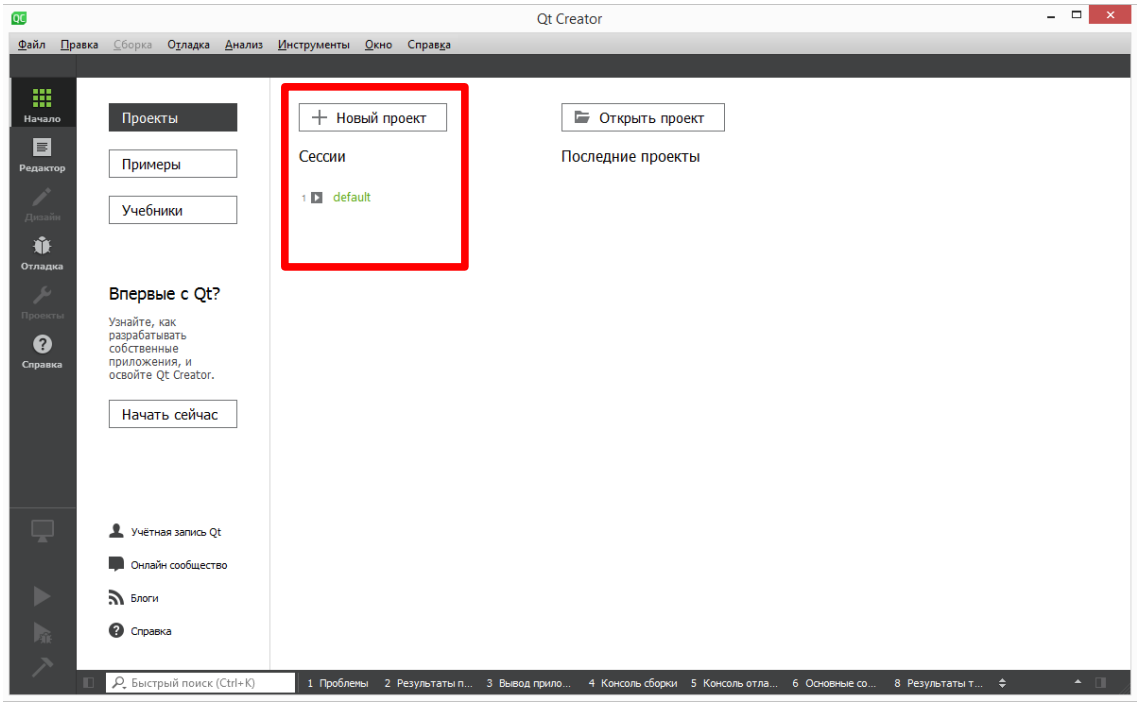

Рисунок 1 Вкладка "Проекты"

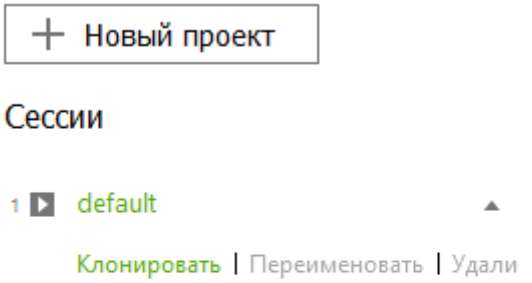

Рисунок 2 Меню сессии

Полноценное управление сессиями, включая создание новых, осуществляется через диалоговое окно, доступное через пункт меню Файл/Сессии/Управление... (рис. 3)

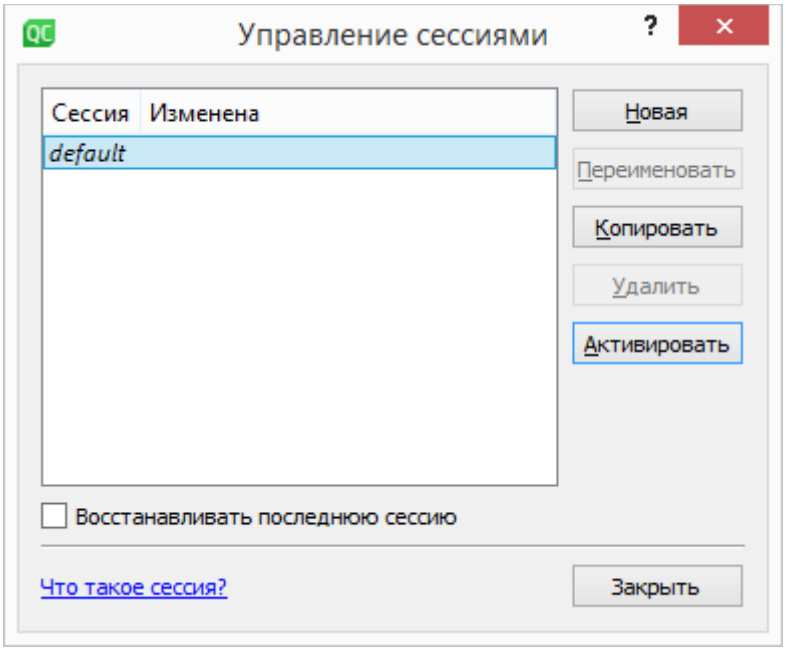

Рисунок 3 Диалог управления сессиями

Создать новую сессию можно, нажав кнопку Новая и указав желаемое имя в появившемся диалоговом окне (рис. 4).

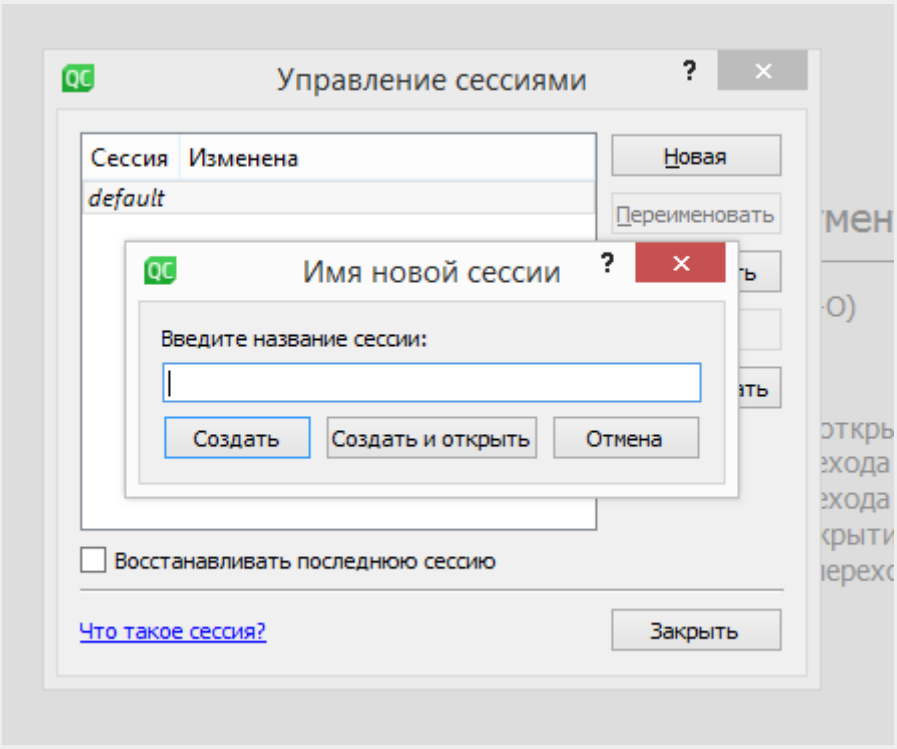

Рисунок 4 Диалог создания новой сессии

Кнопка "Создать и открыть" позволяет сразу активировать созданную сессию.

При успешном выполнении задания в диалоговом окне управления сессиями и во вкладке "Проекты" должна появиться новая сессия (рис. 5-6).

Внимание! Персональную сессию необходимо активировать каждый раз при запуске приложения.

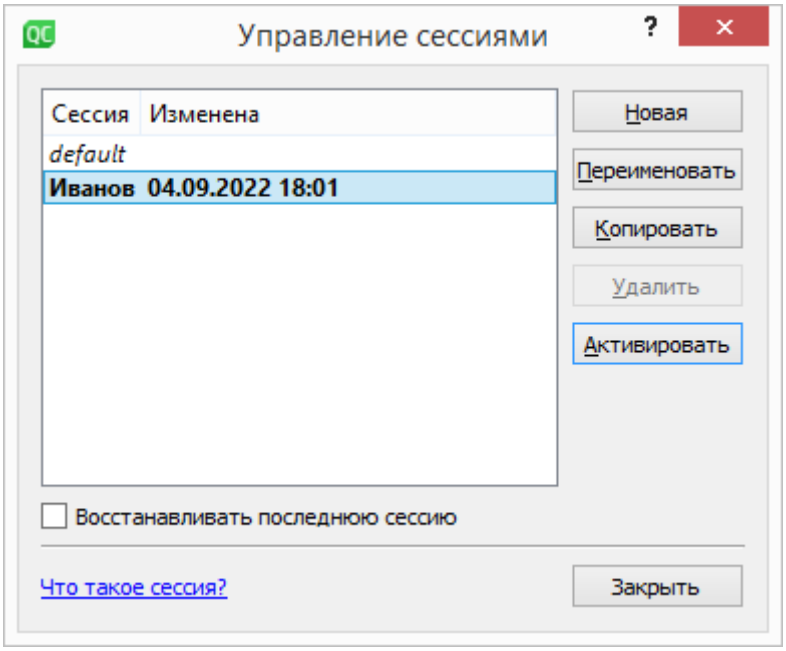

Рисунок 5 Активная пользовательская сессия

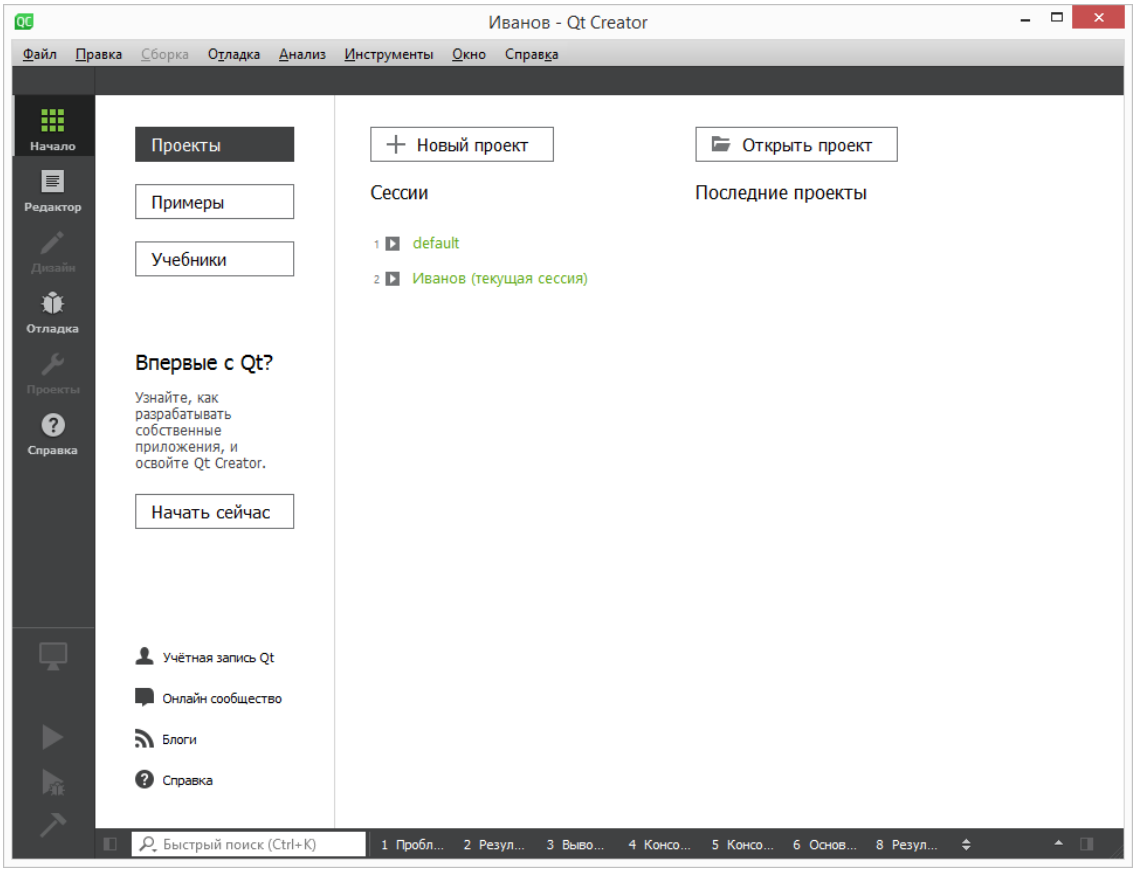

Рисунок 6 Активная пользовательская сессия в панели "Проекты"

Задание 2. Создать новый проект на основе шаблона "Приложение на языке C++".

Новые проекты приложений (и компоненты этих проектов) в QtCreator, как и в большинстве сред разработки, создаются на основе шаблонов конфигурации, настраиваемых в диалоговом режиме (с помощью специального интерфейса — мастера). Диалог выбора шаблона можно открыть, нажав кнопку Новый проект в панели "Проекты", либо активировав пункт меню Файл/Создать файл или проект... (рис. 7). Список доступных шаблонов будет отличаться в зависимости от выбранного способа (в первом случае не будут отображаться разделы для создания файлов и классов). Для запуска мастера необходимо выбрать нужный раздел, затем шаблон, и нажать кнопку Выбрать... (или активировать пункт двойным щелчком).

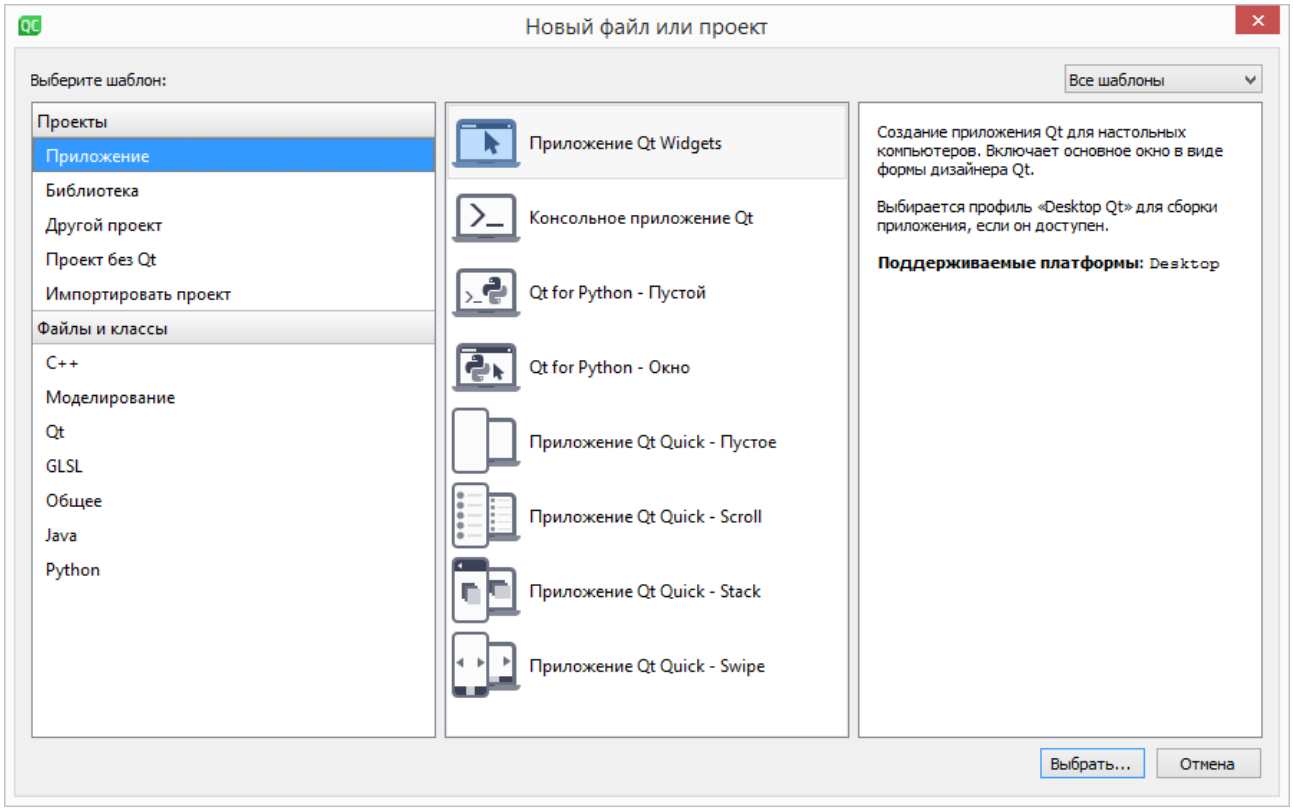

Рисунок 7 Диалоговое окно выбора шаблона

Проект приложения командной строки, не требующего использования библиотек Qt, создаётся мастером "Приложение на языке C++", находящимся в разделе "Проект без Qt" (рис. 8). При успешном выполнении задания должен появиться первый экран мастера (рис. 9).

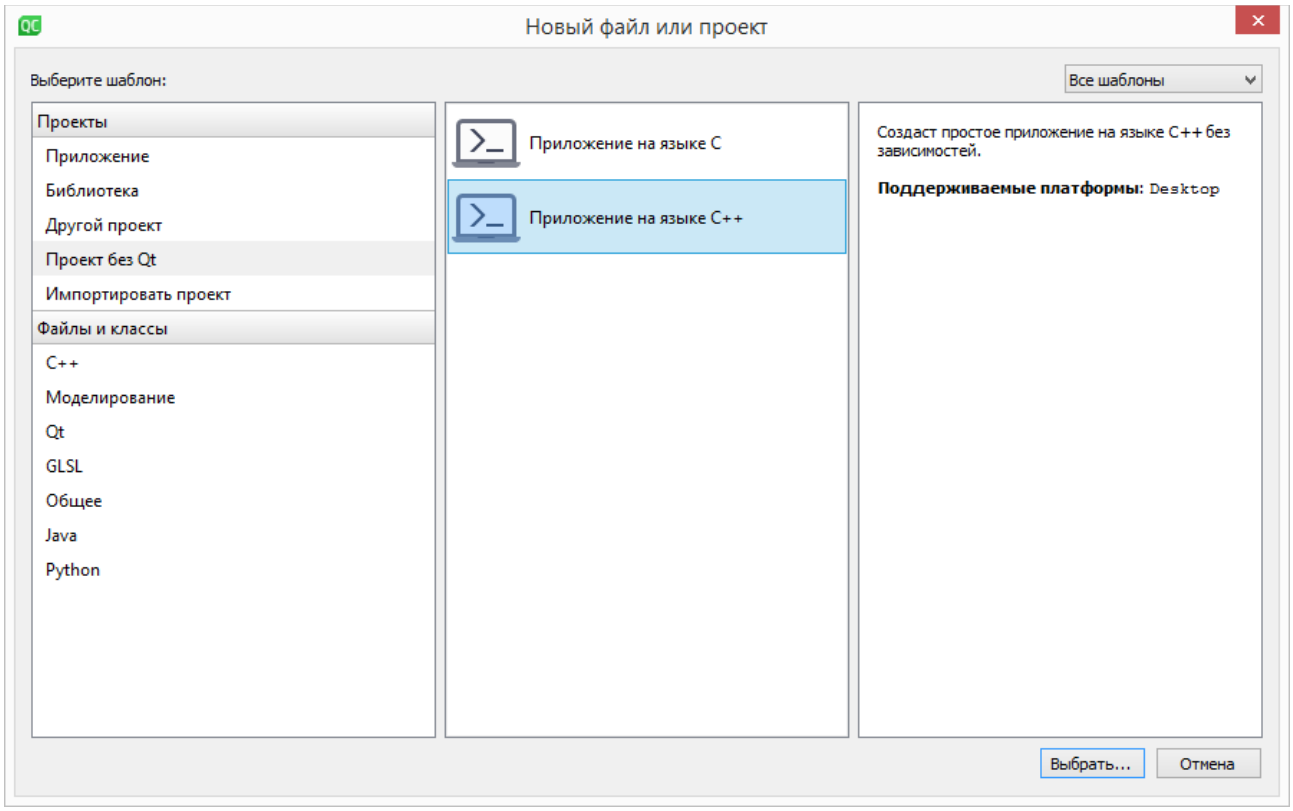

Рисунок 8 Шаблон приложения командной строки

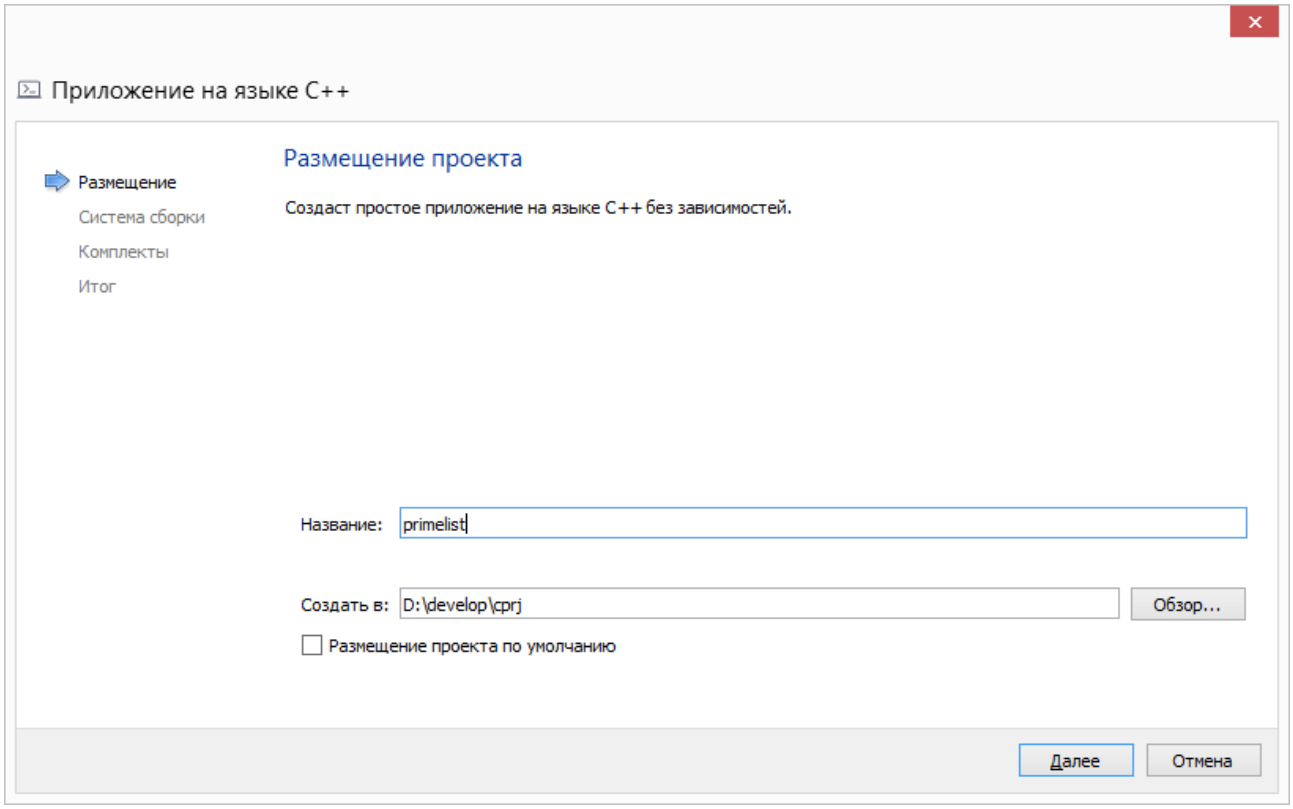

Рисунок 9 Выбор названия и размещения проекта

Задание 2.1. Выполнить настройку проекта с помощью мастера.

На первом экране мастера необходимо указать расположение проекта и его название. При создании проекта будет создан новый каталог с именем, совпадающим с указанным названием проекта, находящийся в каталоге, указанном в поле "Создать в". ,Отметка в поле "Размещение проекта по умолчанию" позволяет запомнить значение базового каталога для всех последующих запусков мастера. Кнопка Далее позволяет перейти на следующий экран мастера.

Внимание! Для корректной работы всех инструментов разработки имя проекта следует выбирать по правилам формирования идентификаторов языка C++ (только алфавитно-цифровые символы из таблицы ASCII и символ подчёркивания, первый символ не должен быть цифрой; другие символы, в т.ч. пробелы — не допускаются).

Аналогичные требования предъявляются к именам всех каталогов базового пути.

Внимание! При работе не на личном ПК, базовый каталог должен располагаться внутри персонального каталога студента. Ставить отметку "Размещение проекта по умолчанию" не следует.

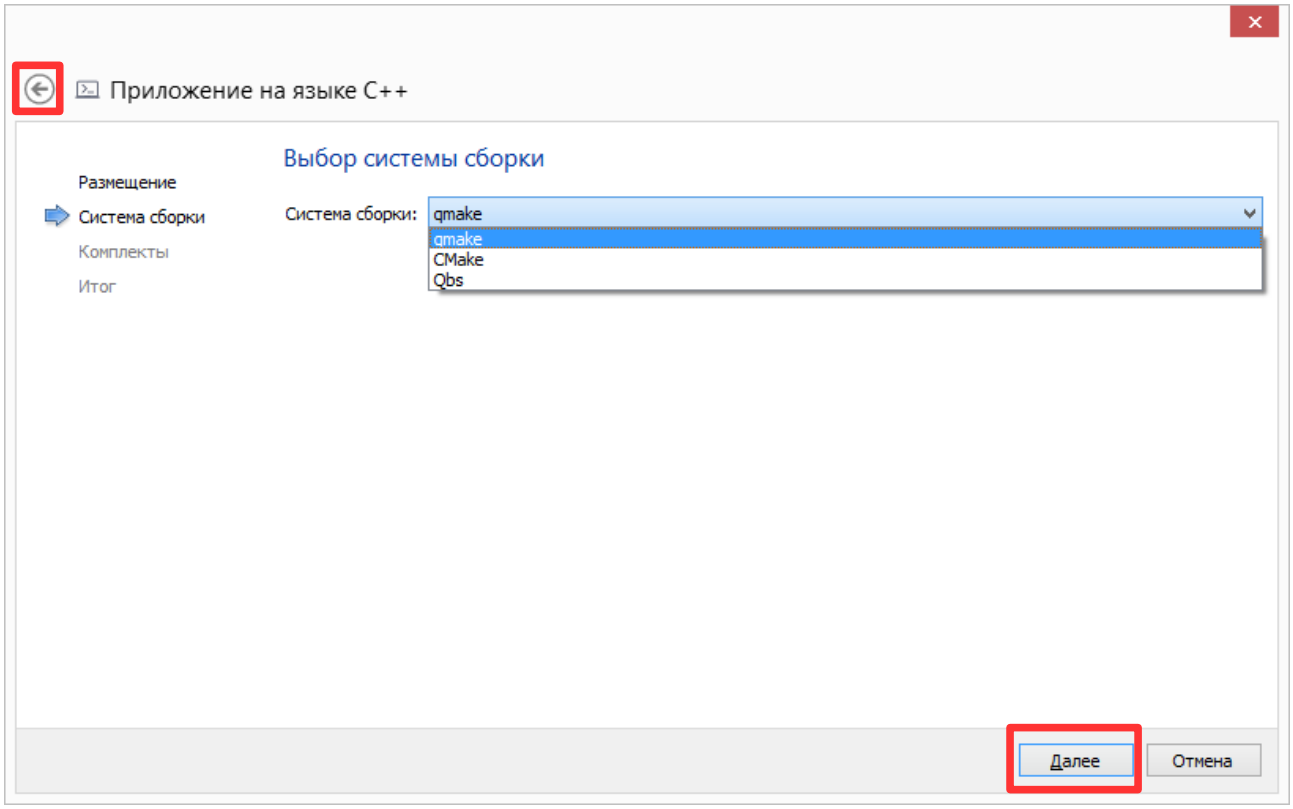

Рисунок 10 Выбор средства управления сборкой

Второй экран мастера содержит выбор программного средства автоматизации сборки. Стандартным для QtCreator является qmake. При необходимости можно вернуться на предыдущий экран , нажав стрелку в левом верхнем углу диалогового окна (рис. 10).

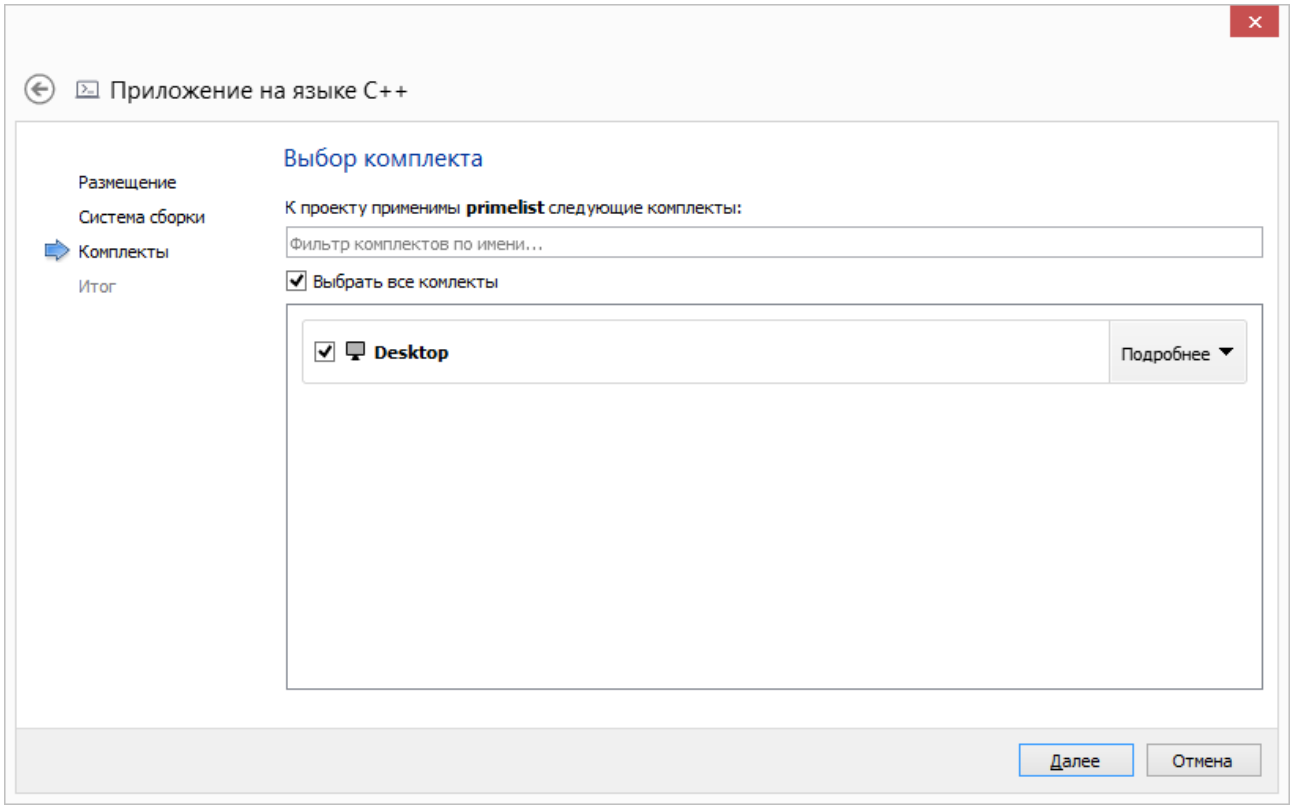

Рисунок 11 Выбор комплекта

На следующем экране выбирается комплект, который будет использоваться для сборки проекта. Комплект — согласованный набор программных инструментов, их конфигураций, и библиотек, используемых средой разработки в процессе создания исполняемых модулей, файлов данных и других компонентов приложения. Комплекты формируются при первом запуске QtCreator на основе сведений, полученных из реестра и переменных окружения ОС, а

также могут быть сформированы вручную через диалог конфигурации среды. Подходящие комплекты должны быть выбраны по умолчанию, поэтому достаточно нажать кнопку Далее.

Внимание! Если все выбранные комплекты являются неполными или по какой-либо другой причине непригодны для создаваемого проекта, кнопка Далее останется неактивной. В этом случае следует прервать работу мастера, нажав кнопку Отмена, и проверить настройки комплектов в диалоге конфигурации среды (при необходимости — обратиться за помощью к преподавателю).

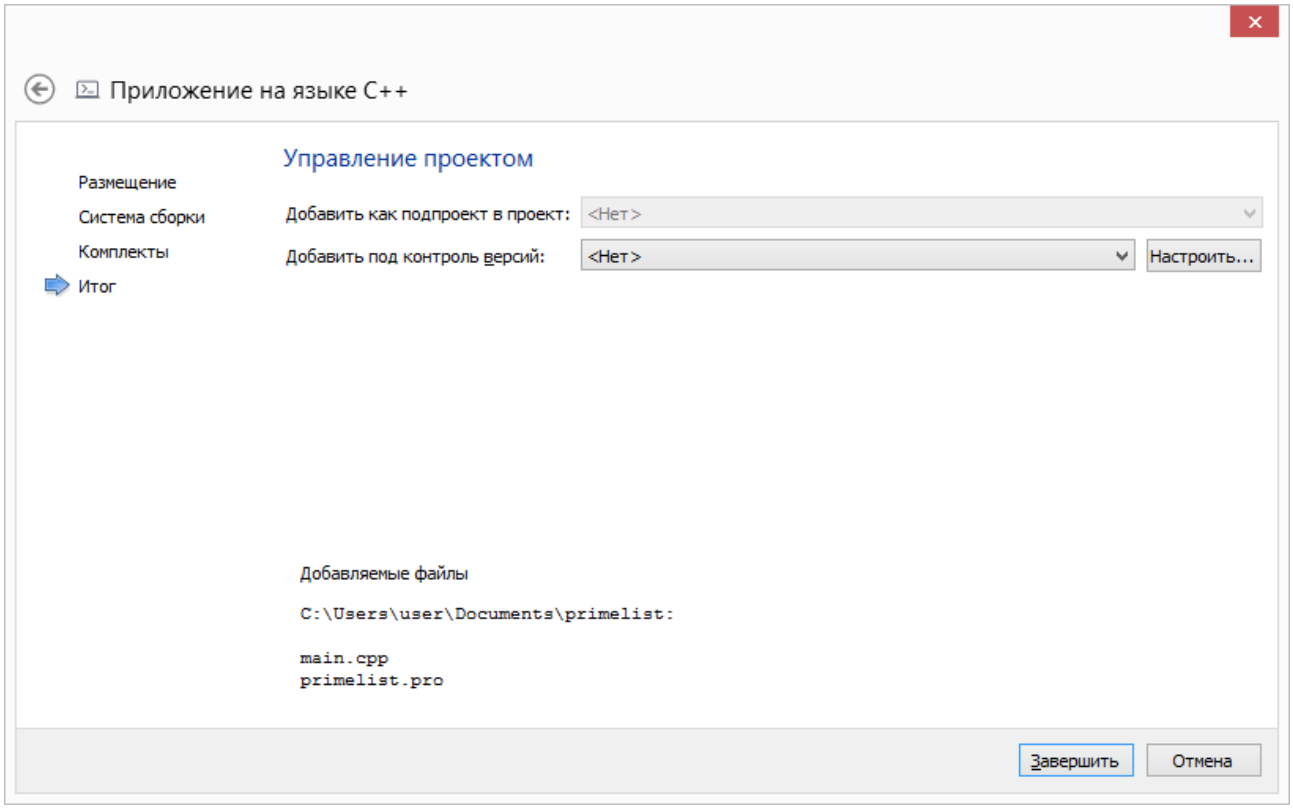

Рисунок 12 Настройки системы управления версиями

Последний этап настройки проекта — настройка связей с другими проектами и выбор системы управления версиями. Поскольку данный проект является самостоятельным и (на данном этапе) не нуждается в контроле версий, в обоих полях следует выбрать "<Нет>". Справочная информация в нижней части этого экрана позволяет проконтролировать расположение и состав файлов проекта (рис. 12).

Задание 2.2. Проконтролировать правильность настроек проекта.

Проект должен располагаться в каталоге, путь к которому состоит из базового и выбранного названия проекта. Путь должен отвечать требованиям, перечисленным в описании соответствующего экрана мастера (см. выше). В проекте должны быть сформированы два файла: собственно файл конфигурации проекта (с расширением .pro) и исходного кода главного модуля (main.cpp).

После проверки корректности всех настроек, работу мастера можно завершить, нажав кнопку Завершить. В результате будет создан каталог проекта, сформированы необходимые файлы, проект открыт и активирован. Интерфейс главного окна среды разработки переключится на вкладку "Редактор".

Вкладка "Редактор" является основной в процессе работы с проектом и, по умолчанию, содержит панели диспетчера проектов (слева), редактора кода (в центре), списка открытых файлов (внизу слева), а также списка текущих проблем (внизу). Интерфейс вкладки может

достаточно гибко настраиваться под потребности пользователя (но на данном этапе это не требуется).

Проект представлен в виде стандартного иерархического списка (дерева). Элемент верхнего уровня представляет сам проект, ниже находятся компоненты проекта. Открыть компонент проекта для редактирования можно двойным щелчком на нём в диспетчере. В зависимости от типа компонента в центральной области откроется окно подходящего инструмента редактирования. Список открытых окон доступен в виде выпадающего списка в верхней части области (рис. 13). Закрыть окно можно кнопкой X непосредственно справа от него.

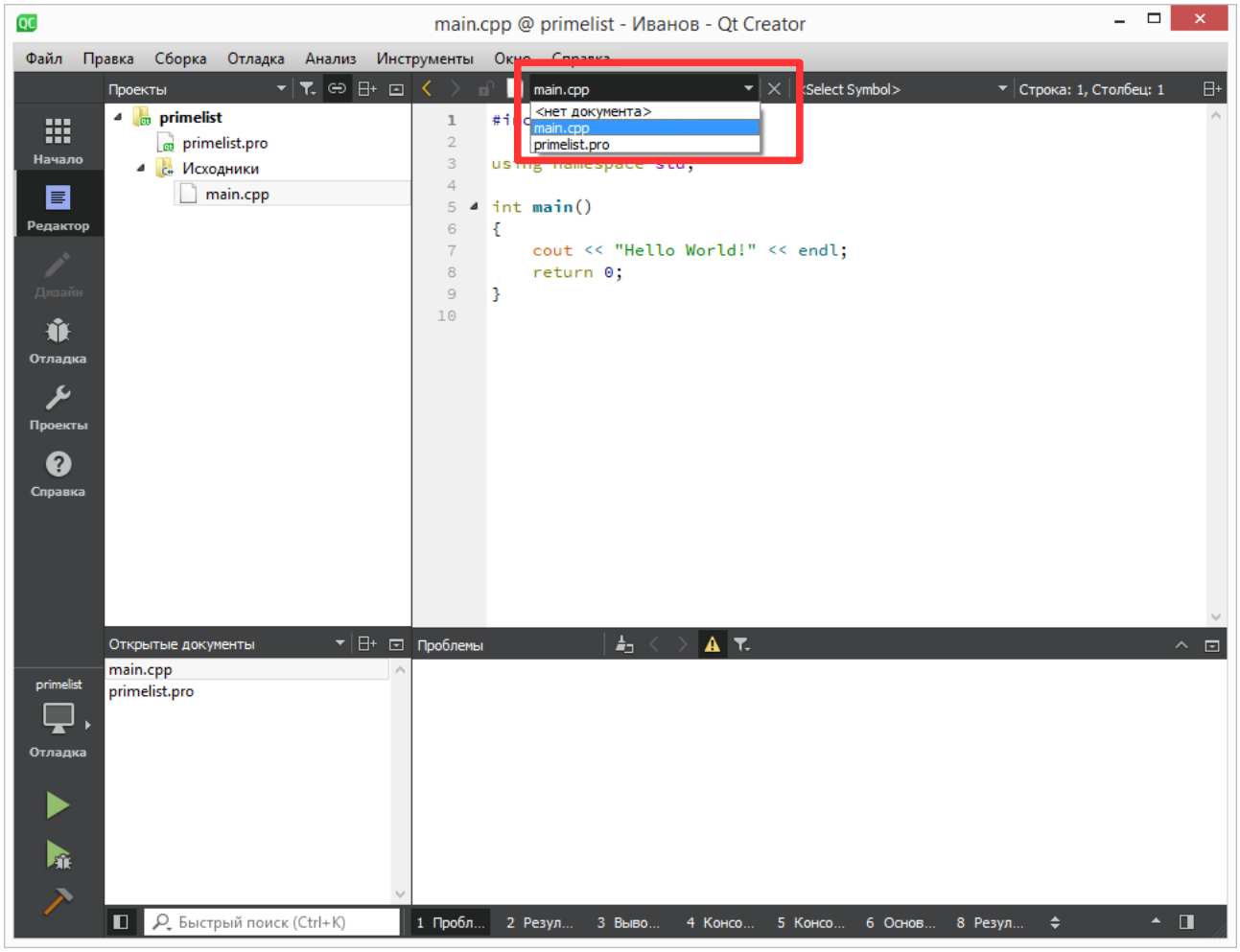

Рисунок 13 Список открытых окон редактирования

В нижней части переключателя панелей находится конфигуратор режимов и кнопки основных действий (запустить, отлаживать, собрать). Конфигуратор режимов позволяет посмотреть текущие настройки сборки и выбрать активную конфигурацию (рис.14).

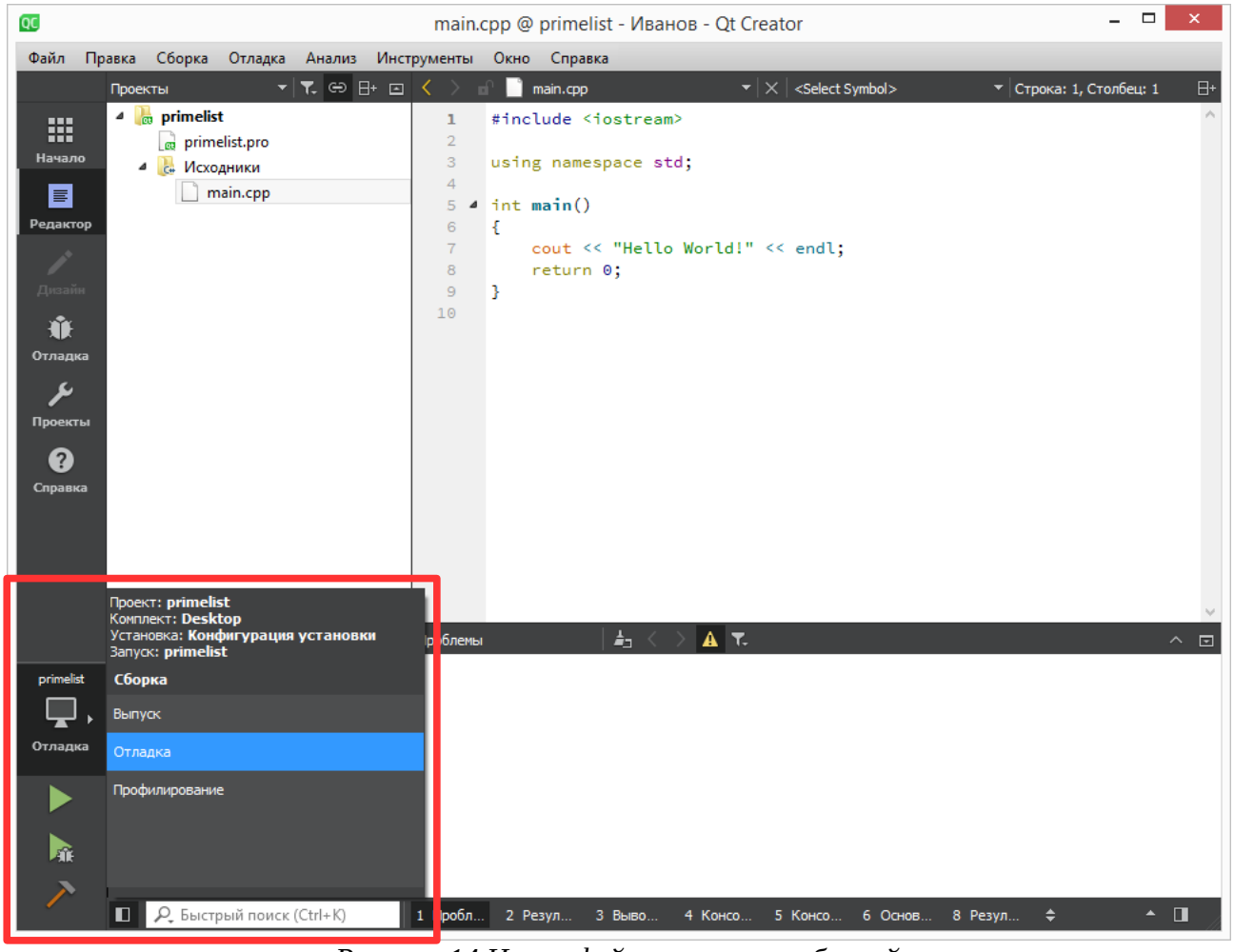

Рисунок 14 Интерфейс управления сборкой

Задание 2.3. Выполнить сборку и запуск проекта для контроля правильности конфигурации.

Для этого необходимо выбрать желаемую конфигурацию и нажать на кнопку запуска. В случае успеха появится окно текстовой консоли с надписью "Hello World!" (рис.15).

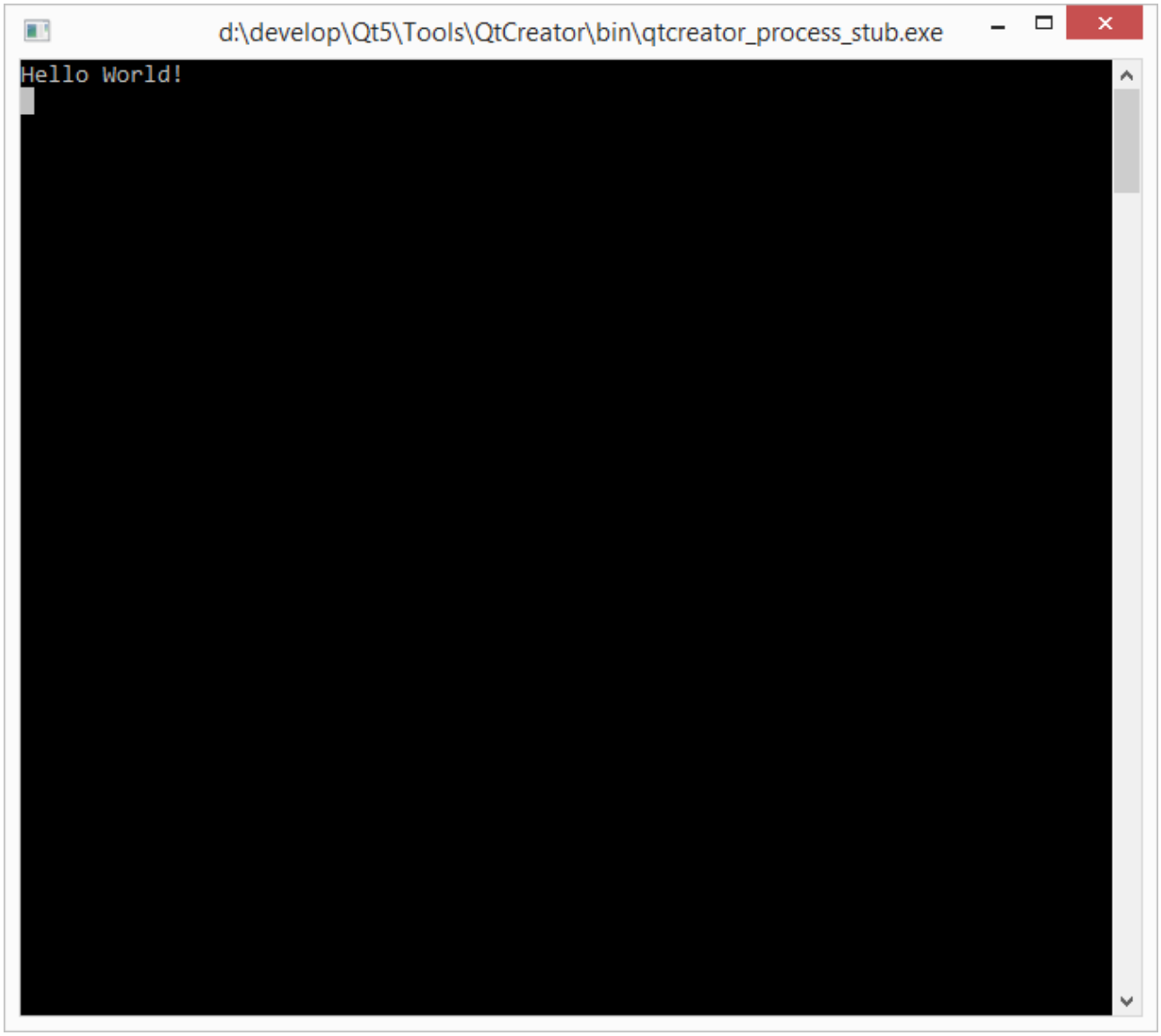

Рисунок 15 Результат сборки и запуска приложения

В процессе сборки в базовом каталоге проекта создаётся каталог сборки, имя которого формируется из названий проекта, используемого комплекта и конфигурации сборки. В каталоге находятся собственно полученные исполняемые модули программы, а также различные вспомогательные файлы (рис. 16). В зависимости от конфигурации, исполняемый файл может находиться в одном из подкаталогов: debug или release. Там же находятся объектные файлы (с расширением .o или .obj) — результат компиляции модулей проекта (рис. 17).

Внимание! Если процесс сборки завершится некорректно, структура каталога сборки и файлов в нём может быть нарушена, что, в свою очередь, может привести к ошибкам при попытках выполнить сборку повторно. В таких случаях рекомендуется удалить каталог сборки, чтобы его структура была сформирована заново.

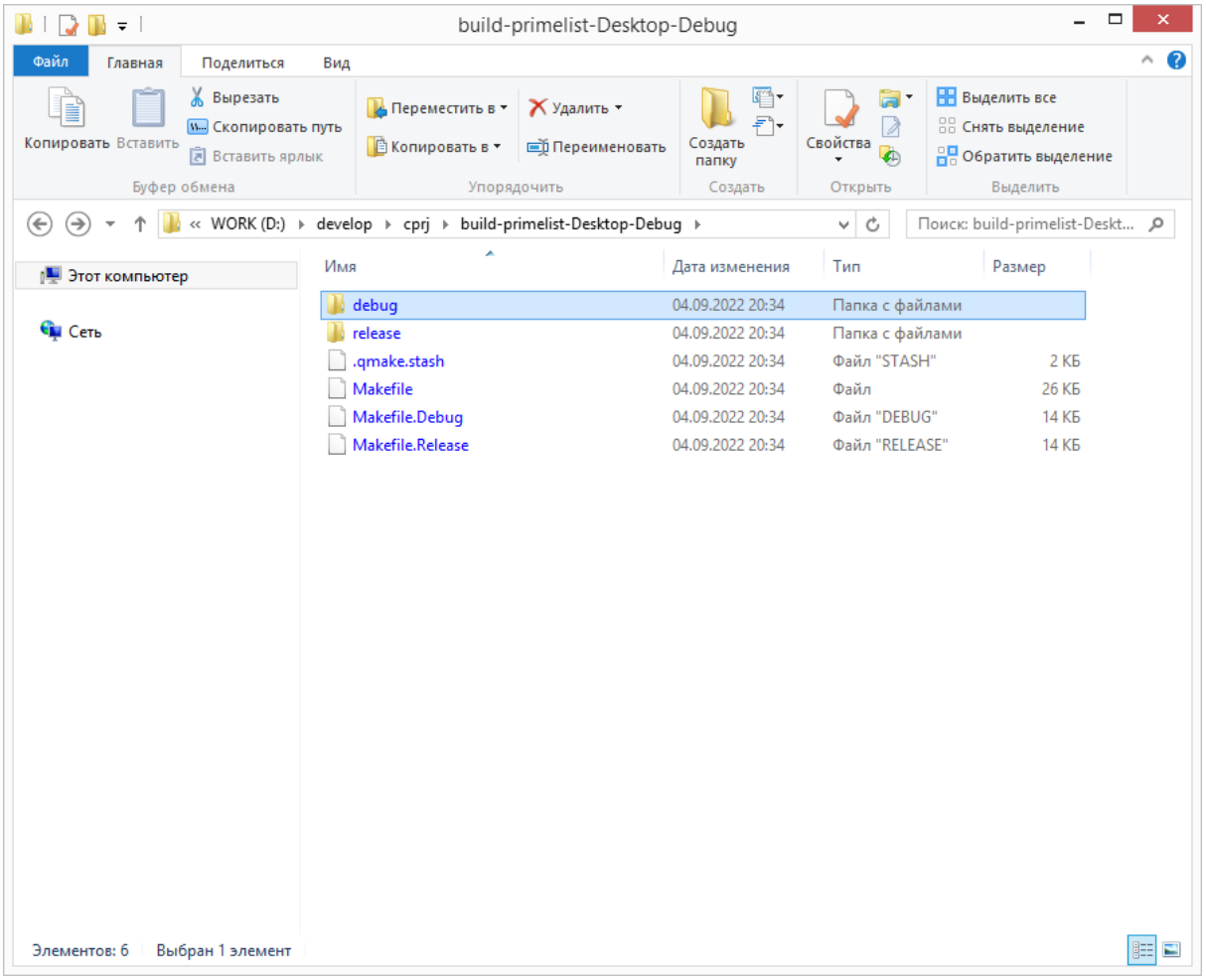

Рисунок 16 Каталог сборки проекта

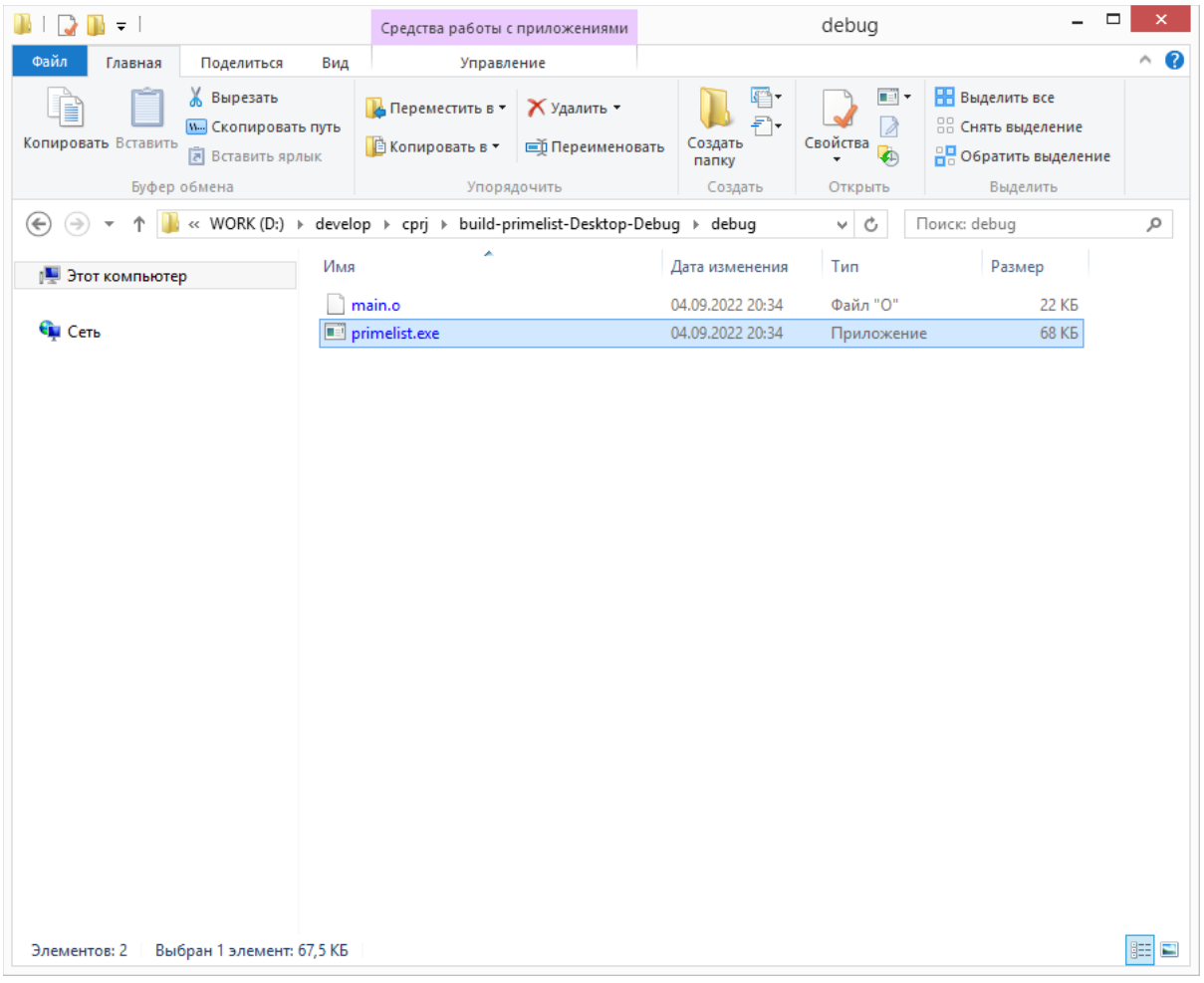

Рисунок 17 Каталог с исполняемым модулем

Задание 3. Реализовать в функции main алгоритм получения от пользователя границ диапазона и вывода последовательности простых чисел, находящихся внутри него.

Примерный код программы приведён на рис. 18. Хотя полученная из этого исходного кода программа успешно решает поставленную задачу, слабая структурированность кода существенно затрудняет дальнейшую работу с ним, а также ограничивает возможности повторного использования для решения похожих задач. Качество кода следует увеличить, выделив в нём структурные компоненты, обладающие, по возможности, самодостаточностью и универсальностью, и оформив их в виде пригодных для эффективного использования структурных единиц языка.

```
#include <iostream>

int main() {
    int left, right;

    std::cin >> left >> right;

    for(int n = left; n <= right; ++n) {
        bool prime = true;
        for(int i = 2; i < n; ++i) {
            if(n % i == 0) {
                prime = false;
                break;
            }
        }
        if(prime) {
            std::cout << n << std::endl;
        }
    }

    return 0;
}
```

Рисунок 18 Пример кода программы

Язык C++ предлагает множество инструментов структурирования кода: функции, классы, пространства имён. Присутствует также ограниченная поддержка модулей , заимствованная у языка C и реализованная при помощи директив препроцессора. Поскольку программа является относительно простой, достаточным будет структурирование на уровне функций и модулей.

Одним из широко применяемых подходов к декомпозиции прикладных программ (и вообще программ, взаимодействующих с пользователем), является отделение интерфейса пользователя от обработки данных, что и будет первым шагом декомпозиции. Дальнейшее структурирование зависит от используемого алгоритма формирования списка чисел. Если используется метод прямого перебора (т.е. цикл с условием), рационально выделить процедуру проверки условия (того, что число является простым).

Задание 4. Выделить процедуры формирования списка простых чисел и (при наличии) проверки числа на простоту в отдельные функции.

Поскольку результатом обработки данных является множество значений, количество которых заранее неизвестно, для передачи их между частями программы необходимо использовать какую-либо динамическую структуру данных. Стандартная библиотека языка C++ предлагает достаточное количество вариантов с различными свойствами (vector, list, set и т.п.). Пример реализации функций приведён на рис. 19.

```cpp
bool isprime(int n) {
    for(int i = 2; i < n; ++i) {
        if(n % i == 0) {
            return false;
        }
    }
    return true;
}

std::vector<int> primelist(int l, int r) {
    std::vector<int> result;
    for(int n = l; n <= r; ++n) {
        if(isprime(n)) {
            result.push_back(n);
        }
    }
    return result;
}

int main() {
    int left, right;

    std::cin >> left >> right;

    auto r = primelist(left, right);
    for(auto n: r) {
        std::cout << n << std::endl;
    }

    return 0;
}
```

Рисунок 19 Пример структурирования кода

Задание 4.1. Выполнить сборку, запуск и проверить работу модифицированной программы.

Поведение программы не должно измениться.

Задание 5. Вынести полученные функции в отдельный модуль так, чтобы в его интерфейсе была только функция получения списка простых чисел.

Модули в языке C++ представляются двумя файлами — заголовком (с расширением .h или .hpp) и реализацией (с расширением .cpp), с (обычно) одинаковыми именами. Соответственно, для добавления в проект модуля необходимо создать два указанных файла.

Задание 5.1. Добавить в проект приложения файлы заголовка и реализации модуля.

В QtCreator необходимую пару можно создать с помощью мастера создания класса. Для этого следует открыть диалог создания с помощью пункта меню `Файл/Создать файл или проект...`, и выбрать шаблон "Класс C++" в разделе "C++" (рис. 20).

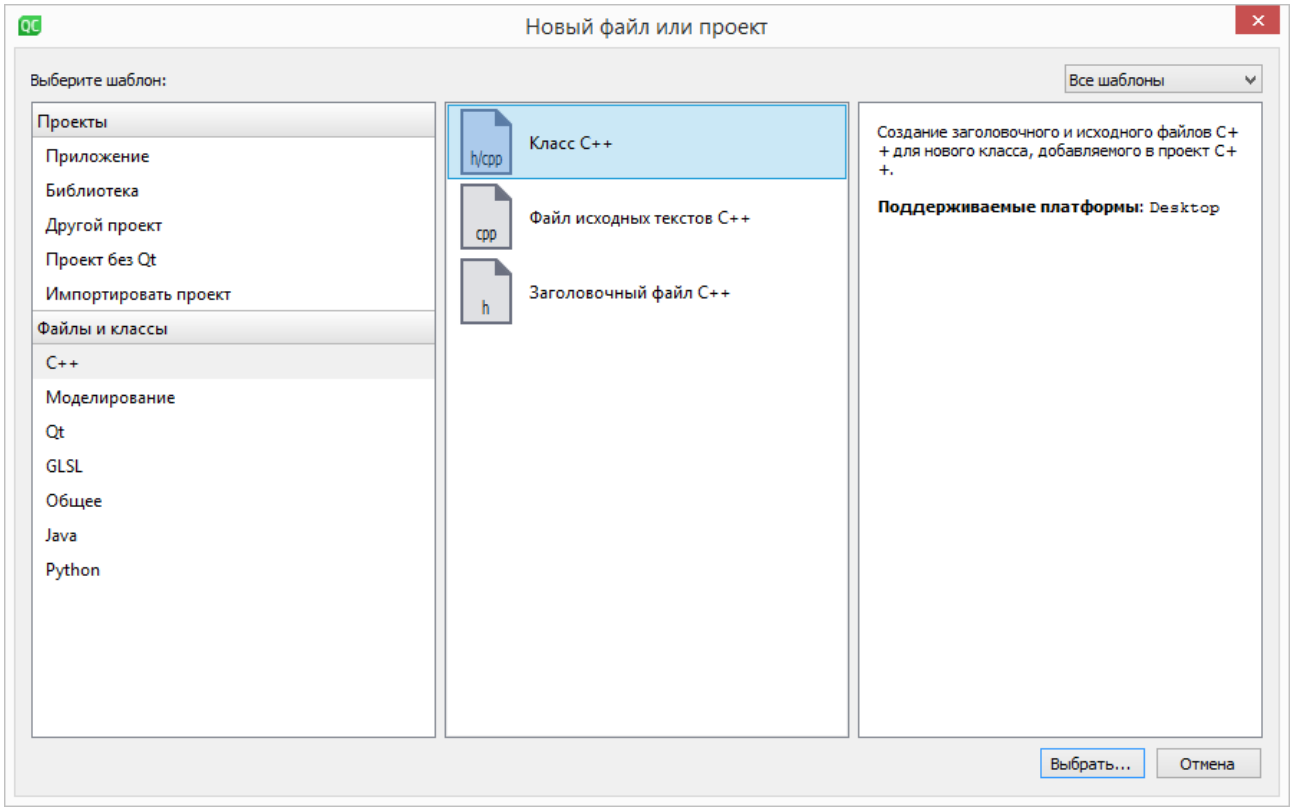

Рисунок 20 Шаблон класса C++

В интерфесе мастера (рис. 21) в качестве имени класса следует указать желаемое имя модуля. Все остальные настройки можно оставить без изменений, т.к. сгенерированный мастером код использоваться не будет. На следующем экране (рис. 22) также достаточно оставить значения по умолчанию. После завершения работы мастера в каталоге проекта будут созданы необходимые файлы и добавлены в состав проекта (рис. 23). Завершающим шагом следует убрать код объявления класса, помещённый мастером в файлы, поскольку структурирование в данном случае делается с использованием функций, а не классов.

> *Внимание! Из заголовочного файла не следует убирать директивы препроцессора. Они нужны для предотвращения дублирования кода при многократном использовании заголовка в других модулях.*

Альтернативный способ получить результат — по отдельности использовать шаблоны "Заголовочный файл C++" и "Файл исходных текстов C++" для создания файлов заголовка и реализации модуля, соответственно.

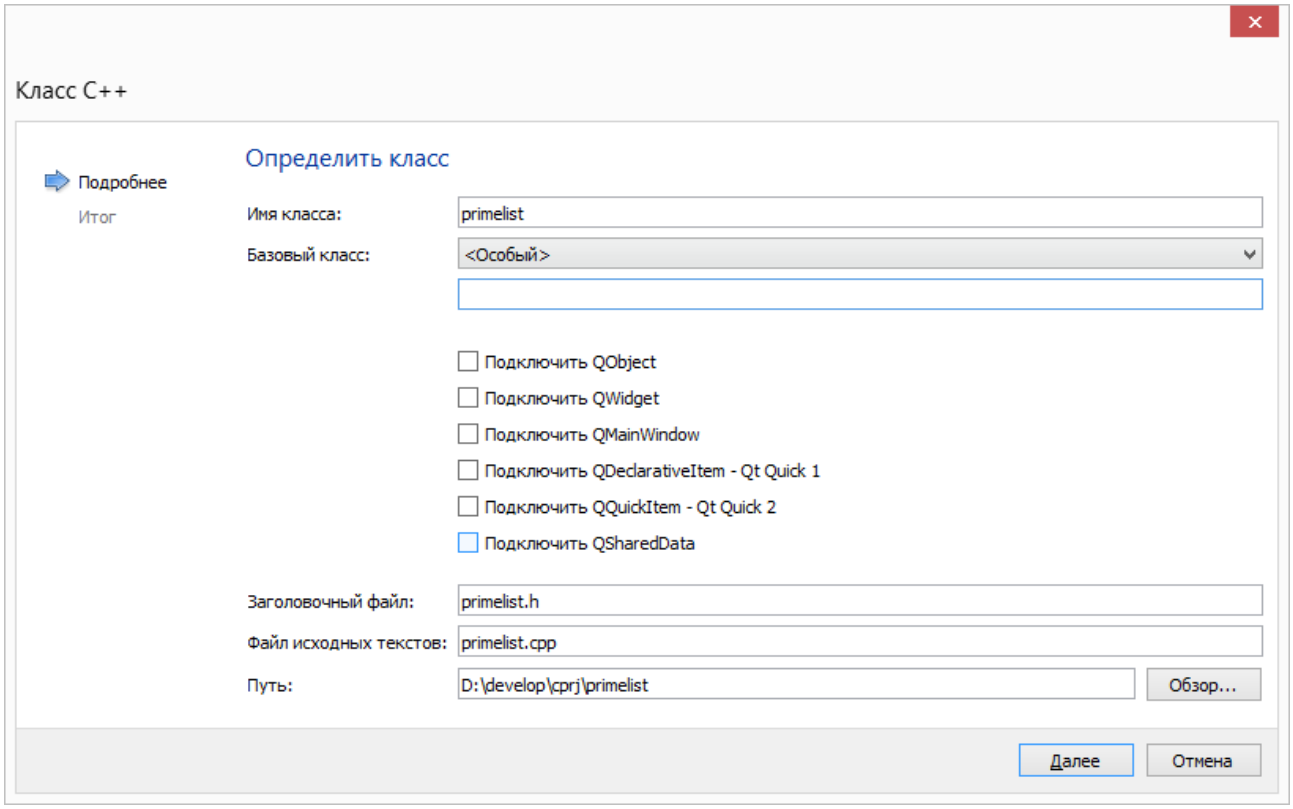

Рисунок 21 Мастер создания класса

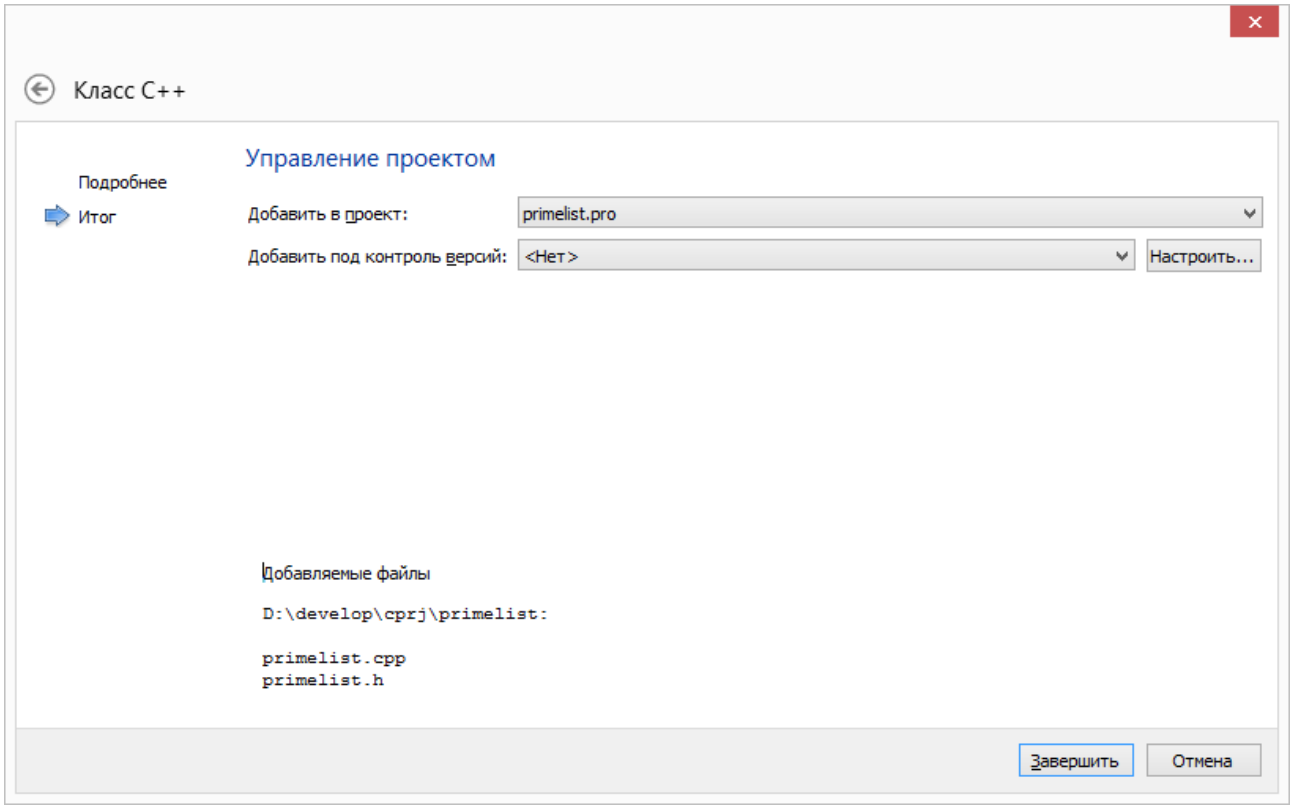

Рисунок 22 Мастер создания класса (продолжение)

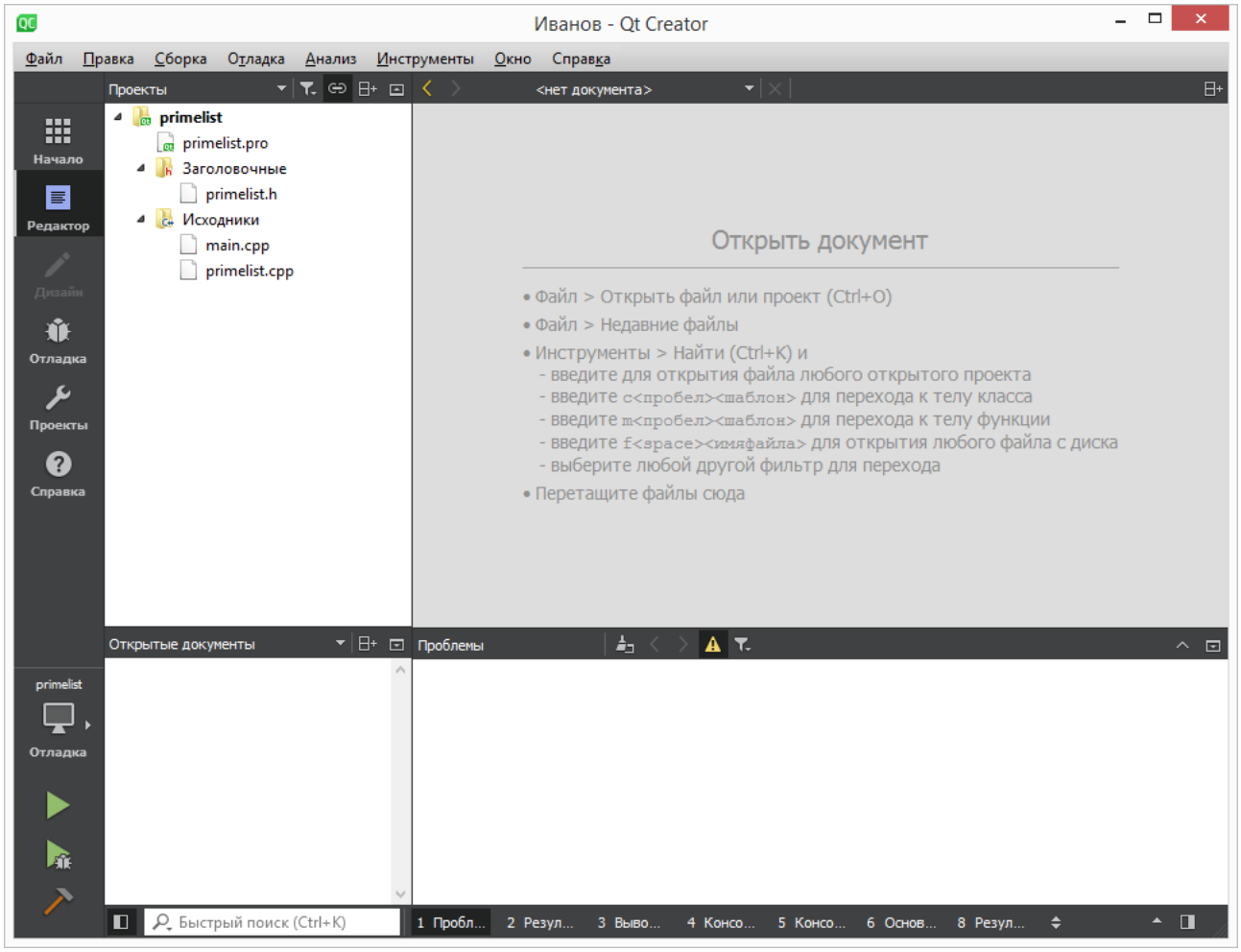

Рисунок 23 Структура проекта после добавления модуля

Задание 5.2. Перенести реализацию функций из главного модуля в созданный.

Для этого достаточно скопировать код функций из файла главного модуля (main.cpp) в файл реализации созданного модуля при помощи стандартных возможностей редактора кода QtCreator по работе с системным буфером обмена. Примерный результат показан на рис. 24.

```cpp
#include "primelist.h"

bool isprime(int n) {
    for(int i = 2; i < n; ++i) {
        if(n % i == 0) {
            return false;
        }
    }
    return true;
}

std::vector<int> primelist(int l, int r) {
    std::vector<int> result;
    for(int n = l; n <= r; ++n) {
        if(isprime(n)) {
            result.push_back(n);
        }
    }
    return result;
}
```

Рисунок 24 Код реализации модуля

Задание 5.3. Добавить в заголовок модуля объявление функции получения списка простых чисел.

Объявление функции в языке C++ повторяет реализацию функции, но вместо тела функции (кода в фигурных скобках) завершается точкой с запятой. Объявление следует поместить между директивой #define и директивой #endif (рис. 25). Поскольку объявление использует имена из стандартной библиотеки, в заголовок модуля следует также поместить соответствующие директивы #include.

```cpp
#ifndef PRIMELIST_H
#define PRIMELIST_H

#include <vector>

std::vector<int> primelist(int l, int r);

#endif // PRIMELIST_H
```

Рисунок 25 Пример кода заголовка модуля

Задание 5.4. Заменить реализацию функций в главном модуле на подключение заголовка модуля обработки данных.

В директиве препроцессора #include файлы могут указываться как в скобках (<...>), так и в двойных кавычках ("..."). В случае отсутствия конфликтов имён модулей эти записи эквивалентны, но существует общепринятое соглашение (и опирающееся на него поведение трансляторов языка при разрешении конфликтов), при котором кавычки используются для "внутренних" заголовков (например, являющихся частью проекта), а скобки — для "внешних" (например, заголовков стандартной библиотеки).

Поскольку созданный модуль является частью проекта, его заголовок в директиве следует указывать в кавычках. Таким образом конечный код главного модуля может выглядеть как на рис. 26.

```cpp
#include <iostream>
#include <vector>
#include "primelist.h"

int main() {
    int left, right;

    std::cin >> left >> right;

    auto r = primelist(left, right);
    for(auto n: r) {
        std::cout << n << std::endl;
    }

    return 0;
}
```

Рисунок 26 Пример кода главного модуля после структурирования

Задание 6. Выполнить сборку, запуск и проверить работу модифицированной программы.