Лекция 2 Microsoft Visual Studio

Microsoft Visual Studio — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию.

Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных. Вы можете создавать:

- приложения и игры, которые выполняются не только на платформе Windows, но и на Android и iOS;
- веб-сайты и веб-службы на основе ASP.NET, JQuery, AngularJS и других популярных платформ;
- приложения для самых разных платформ и устройств, включая, но не ограничиваясь: Office, Sharepoint, Hololens, Kinect и "Интернета вещей";
-игры и графические приложения для разных устройств Windows, включая Xbox, с поддержкой DirectX.
- По умолчанию Visual Studio обеспечивает поддержку C#, C и C++, JavaScript, F# и Visual Basic. Visual Studio хорошо работает и интегрируется со сторонними приложениями.

Visual Studio включает один или несколько компонентов из следующих:

Visual Basic .NET, а до его появления — Visual Basic
Visual C++
Visual C# (включён начиная с Visual Studio .NET)
Visual F# (включён начиная с Visual Studio 2010)
Многие варианты поставки также включают:

Microsoft SQL Server либо Microsoft SQL Server Express
В прошлом в состав Visual Studio также входили продукты:

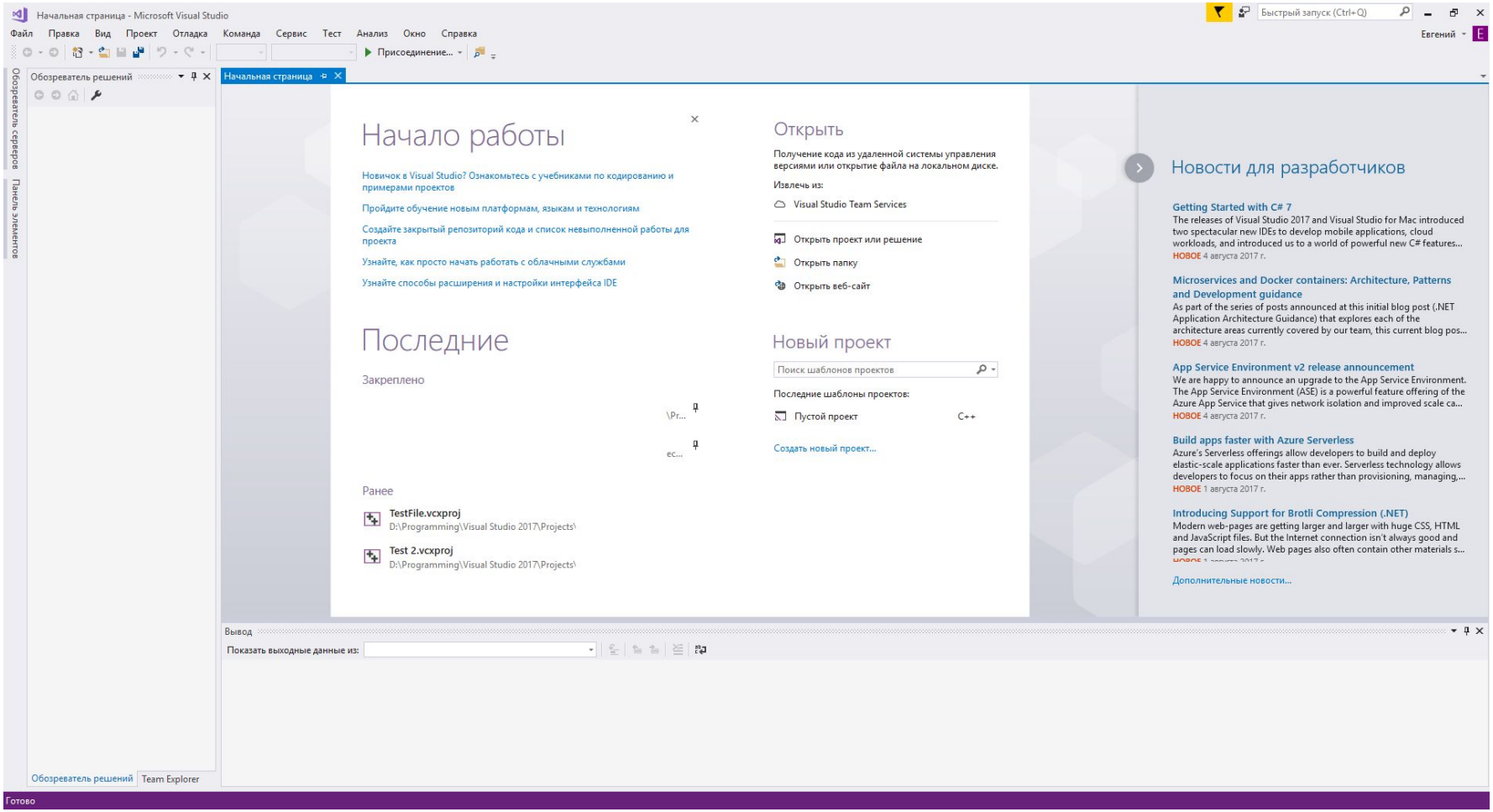
Visual InterDev
Visual J++
Visual J#
Visual FoxPro
Visual Source Safe — файл-серверная система управления версиями
https://www.visualstudio.com/ru/downloads/

https://rdf01.sfedu.ru/RDWeb/Pages/en-US/login.aspx?ReturnUrl=/RDWeb/Pages/en-US/Default.aspx
https://rdf01.sfedu.ru/RDWeb/Pages/en-US/Default.aspx

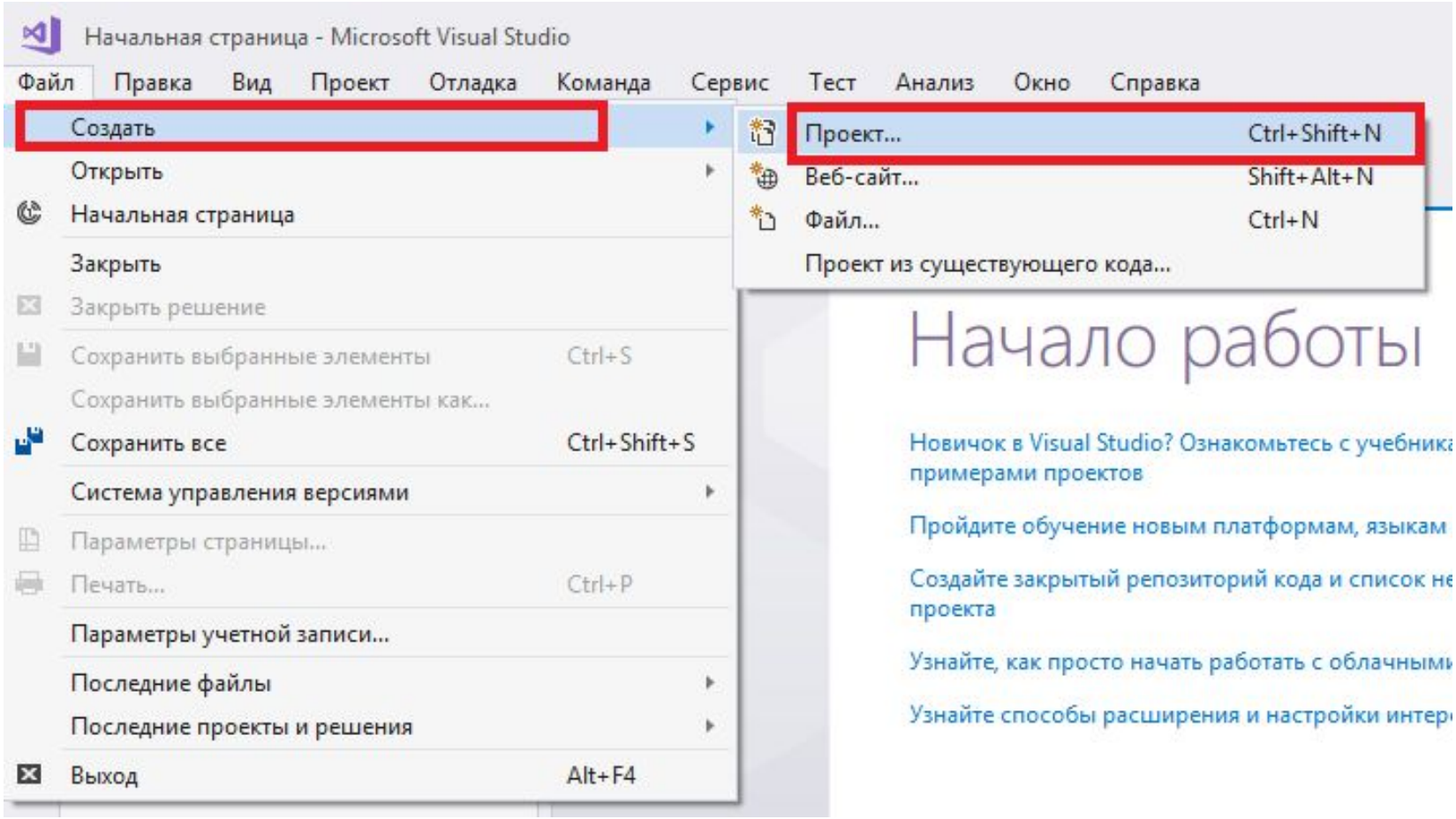
Первая программа на C++ в Visual Studio 2017

1. Создание проекта в Visual Studio

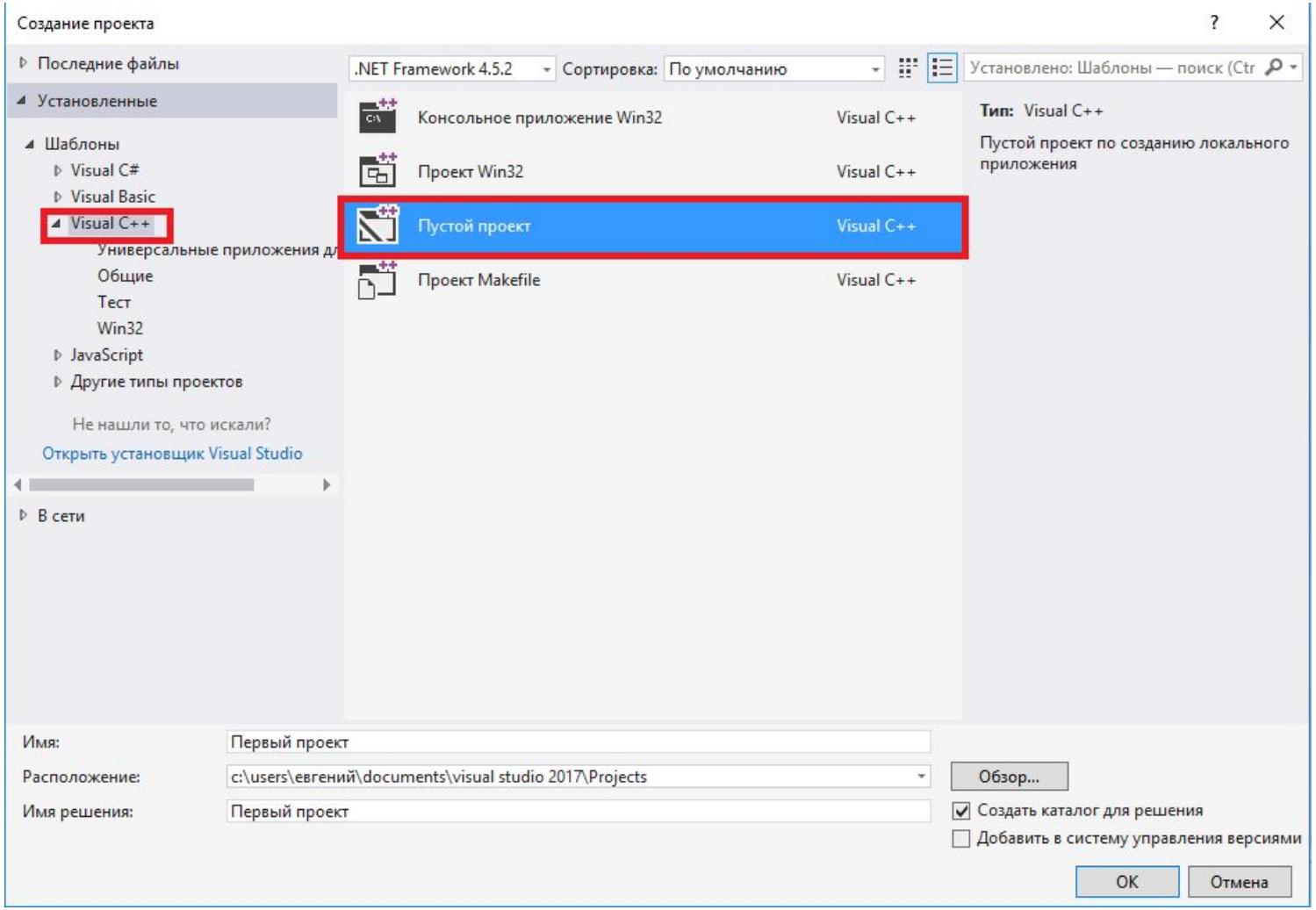
После процедуры установки и запуска Visual Studio откроется вот такое окно:



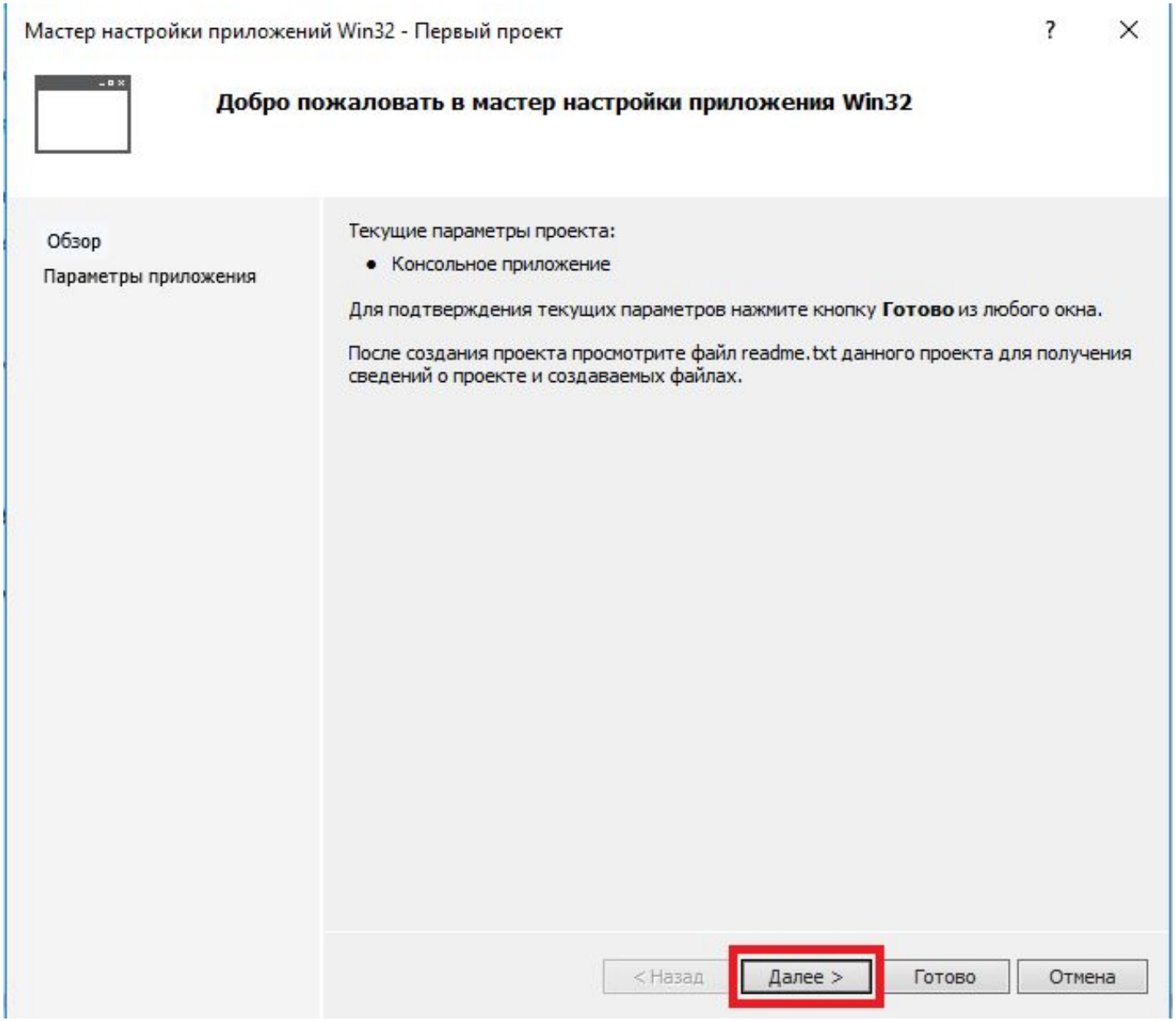
Для начала создадим новый проект. **«Файл-Создать-Проект… (File-New-Project…)»** или сочетание клавиш «Ctrl+Shift+N»



В левом меню выбираем «**Установленные-Шаблоны-Visual C++».** Далее **«Пустой проект»,** прописываем имя нашего проекта, например, «Первый проект», выбираем каталог (папку) в котором будут храниться файлы нашего проекта (программы состоят из нескольких файлов, которые должны быть упорядочены в папке) и нажимаем «ОК»

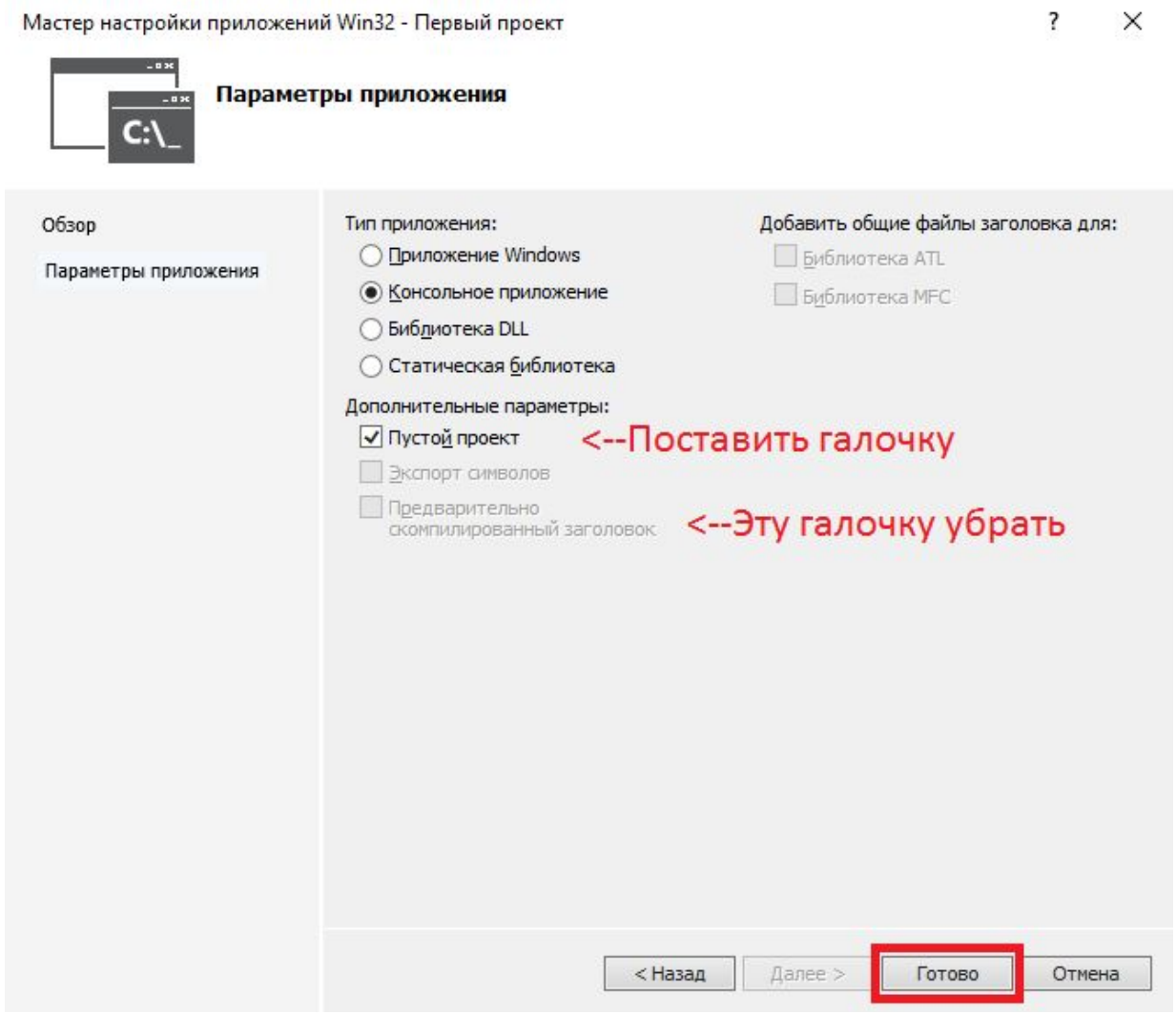


После этого у нас откроется «Мастер настройки приложения». Нажимаем «Далее».



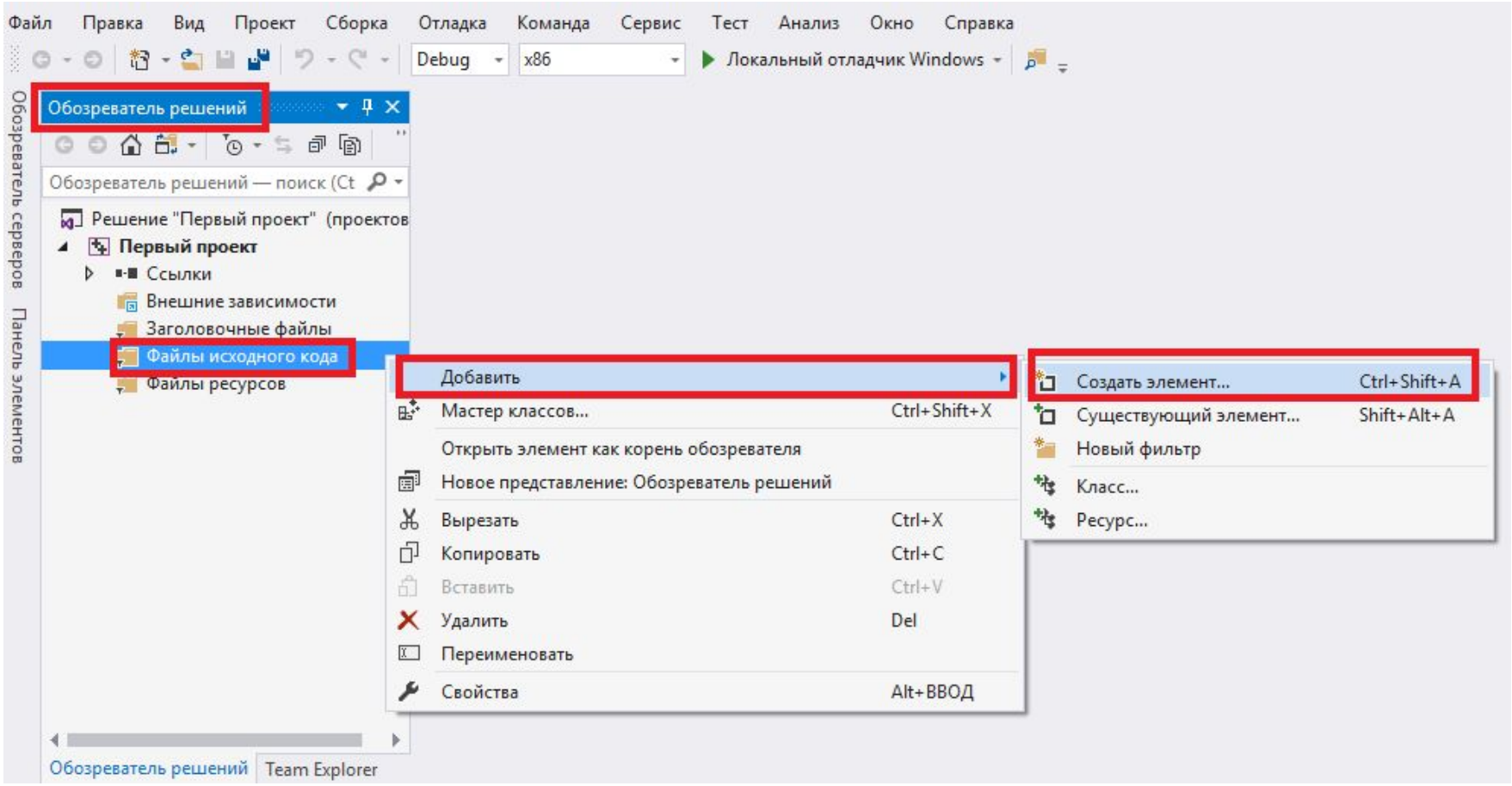
Устанавливаем настройки проекта:

Убираем галочку «Предварительно скомпилированный заголовок»
Ставим галочку «Пустой проект»
Нажимаем «Готово»

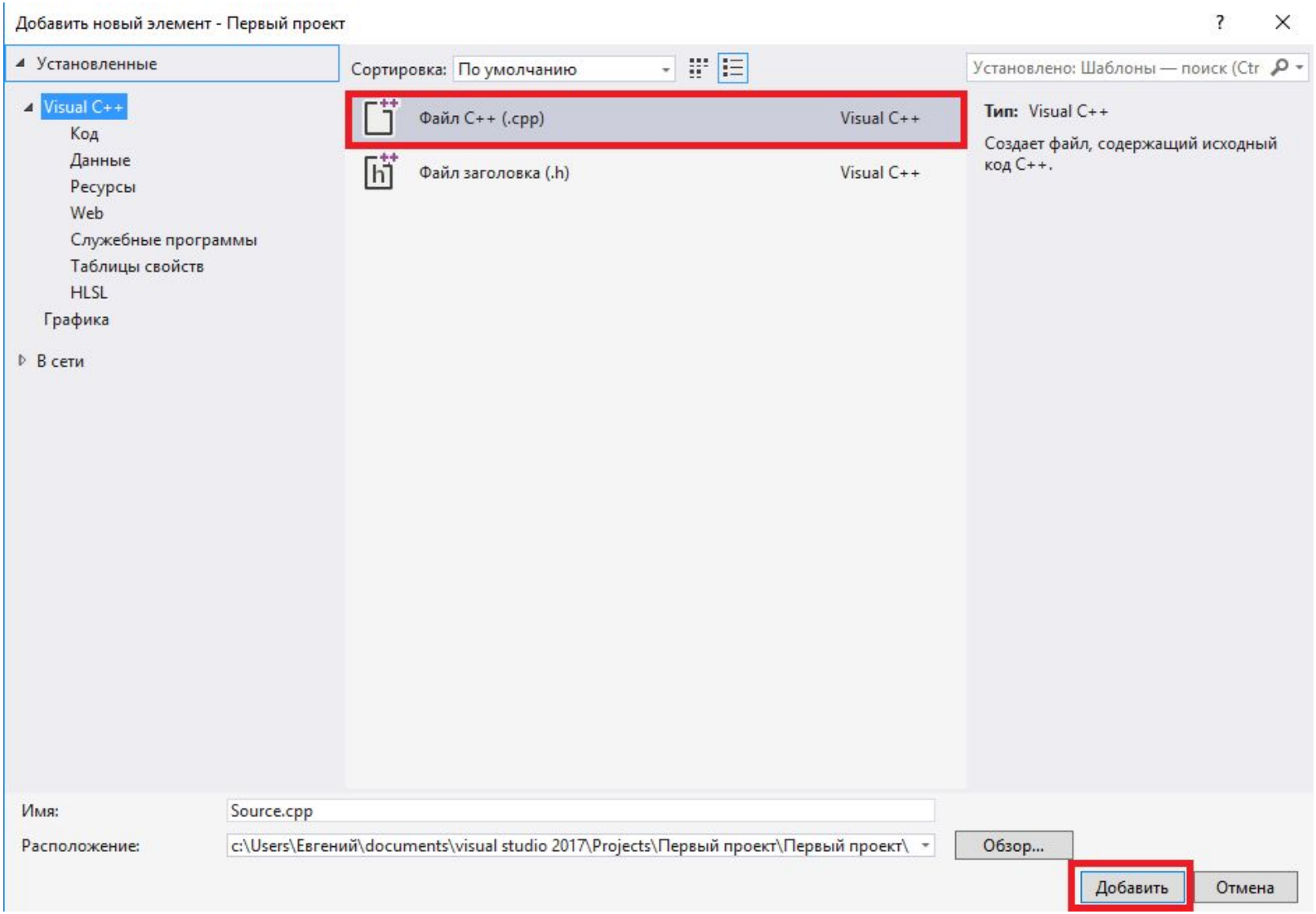


Visual Studio создал пустой проект, в который нам сейчас необходимо добавить файл, в нём мы будем писать программный код **(Файл исходного кода).**

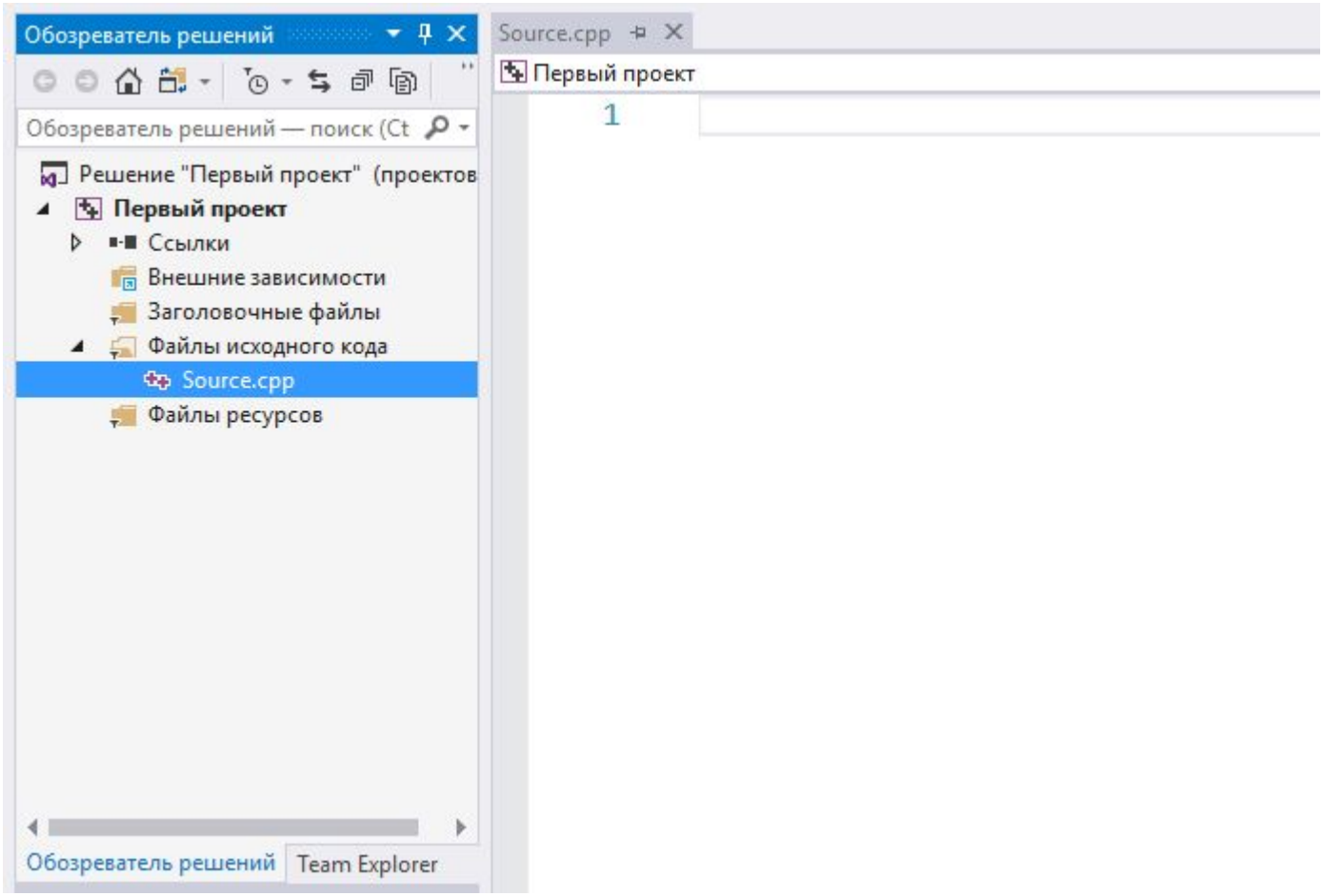
В «Обозревателе решений» правой кнопкой нажимаем на «Файлы исходного кода-Добавить-Создать элемент…» или сочетание **Ctrl+Shift+A**



Выбираем «Файл C++», прописываем его имя, например, «Sourse.cpp» и нажимаем «Добавить».



Теперь наш проект имеет вот такую структуру:



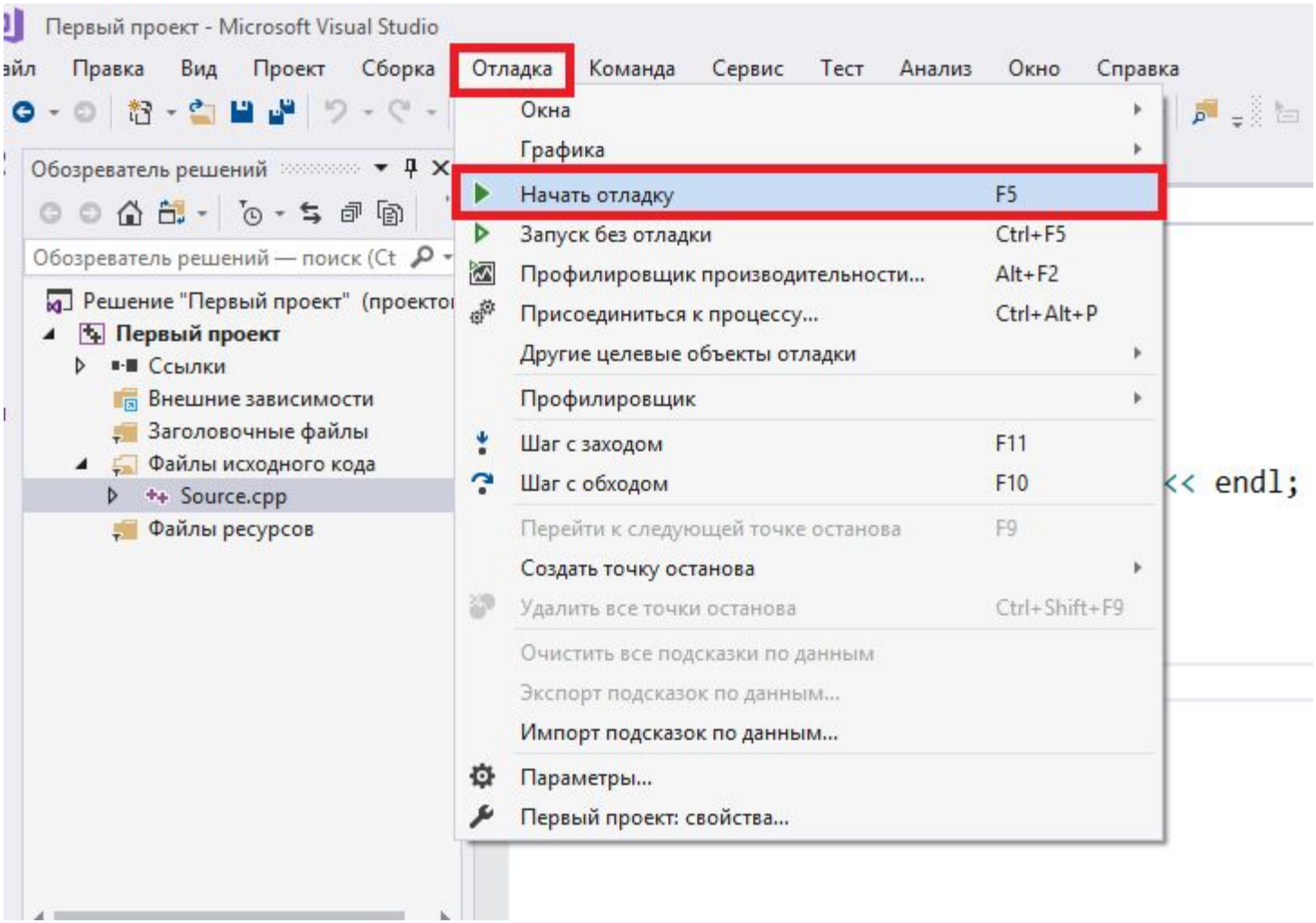
Мы создали первый проект в среде разработки Microsoft Visual Studio, теперь приступаемк написанию кода первой программы.

Запуск первой программы в Visual Studio

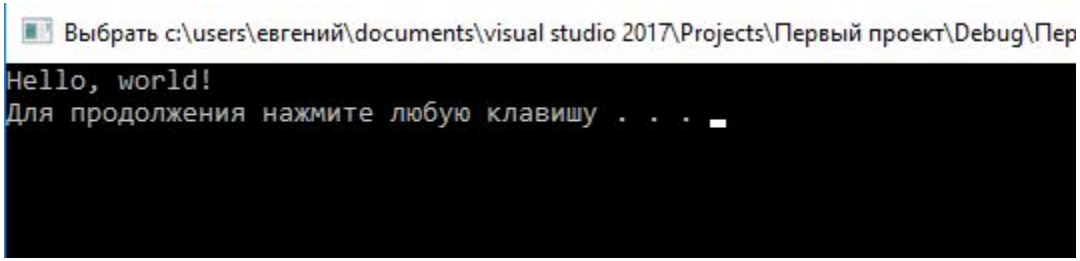
Возьмём для примера простейший программный код из примера **Первая программа на C++**.

```
1  #include <iostream>
2
3  int main()
4  {
5      using namespace std;
6
7      cout << "Hello, world!" << endl;
8      system("pause");
9
10     return 0;
11 }
```

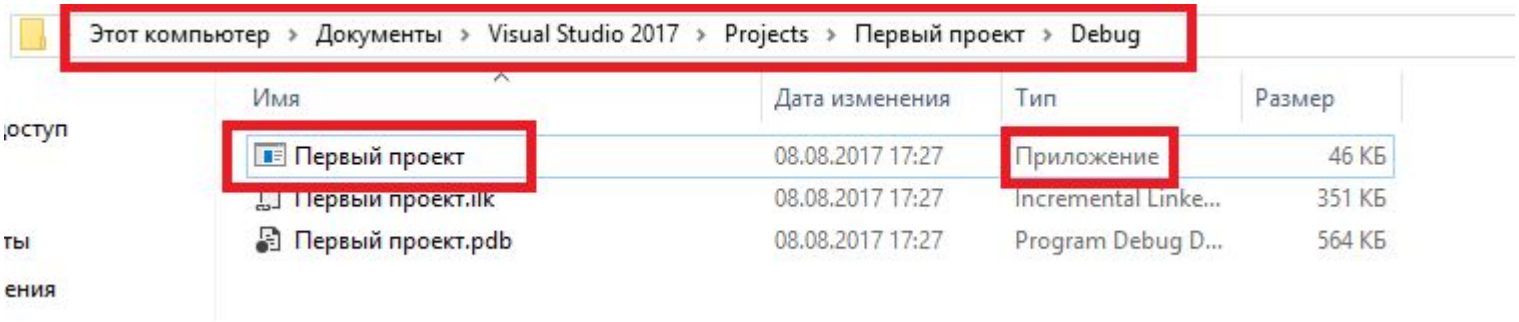
Для запуска программы необходимо нажать «Отладка-Начать отладку», либо клавиша «F5»



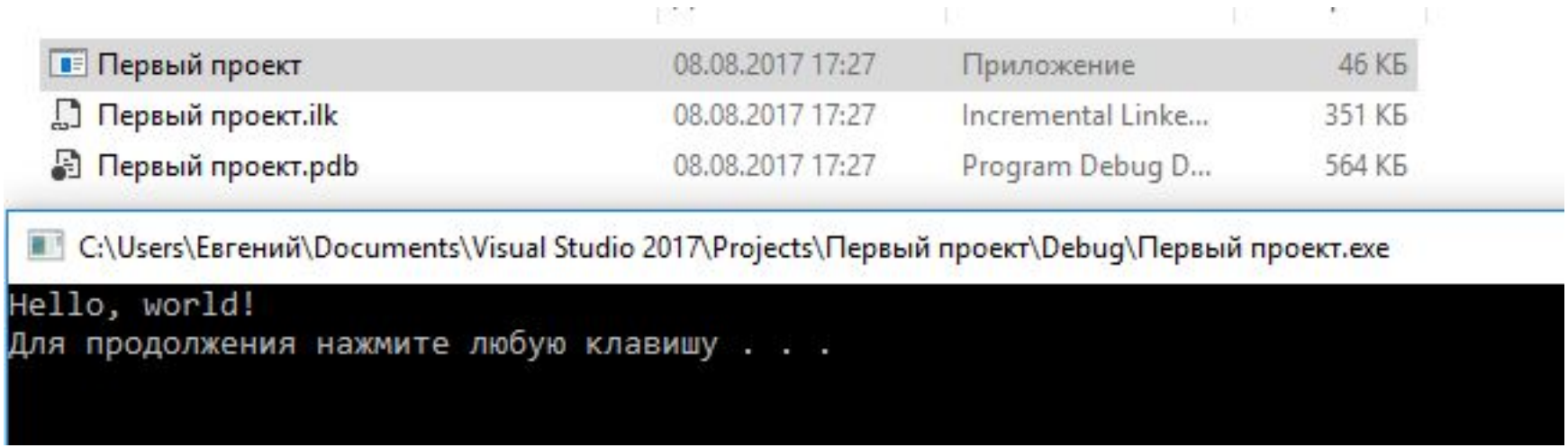
В командной строке запускается наша программа.



Visual Studio создал файл с форматом .exe, теперь Вашу программу можно запускать и на других компьютерах.
Чтобы найти этот .exe файл идем в папку с нашим проектом, в папке «Debug» и находим файл «имя_проекта.exe», в моём случае это «Первый проект.exe»
Вот так выглядит полный адрес до программы, и папка в котором она находится:



В любой момент можно запустить программу из этой папки без использования Visual Studio.



Код программы:

```
1  #include <iostream>
2
3  int main()
4  {
5      using namespace std;
6
7      cout << "Hello, world!" << endl;
8      system("pause");
9
10     return 0;
11 }
```

В строчке 1:

1 #include <iostream>

include называется директивой предпроцессора, с помощью неё можно подключить различные библиотеки функции. Дело в том, что в языке программирования C++, чтобы использовать некую функцию, необходимо предварительно подключить библиотеку, которая содержит данную функцию, и это необходимое условие!

iostream – библиотека, отвечающая за вывод данных.

Программистами написано множество функций: для вывода данных на экран, математические функции (sin, cos, tan и т.д.), функции работы с файлами, текстом и т.д.

Чтобы использовать все эти функции в своей программе необходимо подключить библиотеку, которая содержит уже написанные функции с помощью директивы предпроцессору **#include.**

В нашей программе мы подключили библиотеку поточного ввода-вывода ***iostream.***

В Visual Studio предлагается огромное количество уже готовых библиотек.

Математические функции — sin, cos, tan содержаться в библиотеке **cmath**;

Функции вывода информации на экран cout, printf в **iostream**;

Функции работы с файлами — fopen, rename, remove в **cstdio.**

Строчка 5:

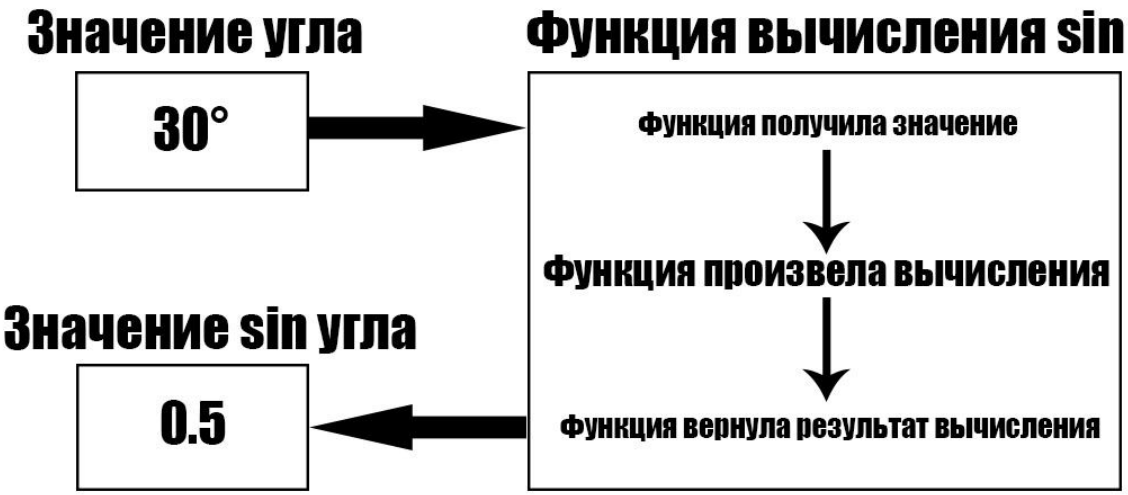
1 int main()

Это главная функция программы. Именно с неё начинается выполнение всей программы. Она может содержать программный код или другие функции. Главная функция имеет вид:

```
1 int main()
2 {
3
4     return 0;
5 }
```

Здесь **int** обозначает тип возвращаемого значения (Типы данных в C++). Любая функция (исключением является функция с типом **void**) должна возвращать значение.

У нас есть математическая функция, которая вычисляет синус угла. Передав в функцию значение угла, мы получим значение синуса этого угла. Так вот, синуса угла — это и есть возвращаемое значение функции. Возвращаемое значение функции — это значение, которое функция вернула после выполнения своей работы (в нашем случае вычисления).



Функция возвращает значение через оператор return.

Пример:
return 0; — функция возвращает значение 0.
return -215; — функция возвращает значение -215.
return 3.1415926; — функция возвращает значение 3.1415926.

Функция может возвращать код ошибки, если её работа была некорректна или сообщение об отсутствии ошибок, если все прошло по плану.
В C++ принято, что если функция вернула **значение 0**, то она завершилась без ошибок.
Иначе мы можем прописать код ошибки для каждого случая, когда функция аварийно завершила свою работу. В зависимости от кода ошибки мы можем понять, что не так в работе функции.

Предположим, что код ошибки 1 (return 1) – значит у нас **нет доступа к процедуре созданию файла на жестком диске.**
Код ошибки 2 (return 2) **– имя файла превышает установленную длину**. Для каждого случая можно указать свой код ошибки, чтобы было удобнее работать с функцией и понимать, что за ошибку она выдала после некорректного завершения работы.
В круглых скобочках main() могут содержаться параметры при запуске программы через командную строку (**Аргументы командной строки**). Нам пока они не нужны, поэтому скобки и пустые.

Фигурные скобки { } обозначают **тело функции.** В теле функции содержится тот программный код, который должен выполняться при запуске программы.

Строка 3:

```
1  using namespace std;
```

Обозначает, что в программе мы используем стандартное пространство имен. Дословно «использовать пространство имен std».

Пространство имен — это декларативная область, в рамках которой определяются различные идентификаторы (имена типов, функций, переменных, и т. д.). Пространства имен используются для организации кода в виде логических групп и с целью избежания конфликтов имен, которые могут возникнуть, особенно в таких случаях, когда база кода включает несколько библиотек. Все идентификаторы в пределах пространства имен доступны друг другу без уточнения. Идентификаторы за пределами пространства имен могут обращаться к членам с помощью полного имени для каждого идентификатора

На начальных этапах Вам будет достаточно запомнить, что в программе следует использовать стандартное пространство имен, то есть перед главной функцией прописывать using namespace std.

Строка 7:

```
1  cout << "Hello, world!" << endl;
```

С помощью **cout** выводится сообщение, заключенное в кавычках. Манипулятор **endl** переводит курсор на новую строку. Если бы мы не прописали endl, то следующее сообщение было бы слито с предыдущим в одной строке. Самостоятельно вы должны попробовать вывести на экран несколько строк, с использованием манипулятора endl после каждой строки и без его использования. Вы сразу же поймете зачем нужен этот манипулятор.

Строка 9:

```
1 system("pause");
```

Функция system передает команду, заключенную в кавычках, командному процессору.
Если в кавычках мы передадим команду pause, то окно консоли не закроется, пока вы не нажмете любую клавишу.
Попробуйте запустить программу с system(«pause»); затем сотрите эту строку и посмотрите, как будет вести себя программа.

Выводы

- Директива предпроцессора include подключает библиотеки с функциями в Вашу программу.
- Для работы необходимо использовать стандартное пространство имён (using namespace std;)
- Выполнение программы начинается с главной функции main. То, что в фигурных скобках { } называется телом функции, и там содержится программный код.
- Вывод сообщения на экран производится с помощью cout, а форматирование с помощью endl;
- Функции имеют тип возвращаемого значения. Функция возвращает значение (результат её работы) через оператор return.

Задание:
Сделать визитку о себе (ФИО, город проживания, e-mail, skype).

Она должна иметь вид: