Лабораторная работа 1

**Знакомство со средой программирования Visual Studio 2017**

***Упражнение 1.*** Создание проекта консольного приложения.

**1. Создание проекта консольного приложения**

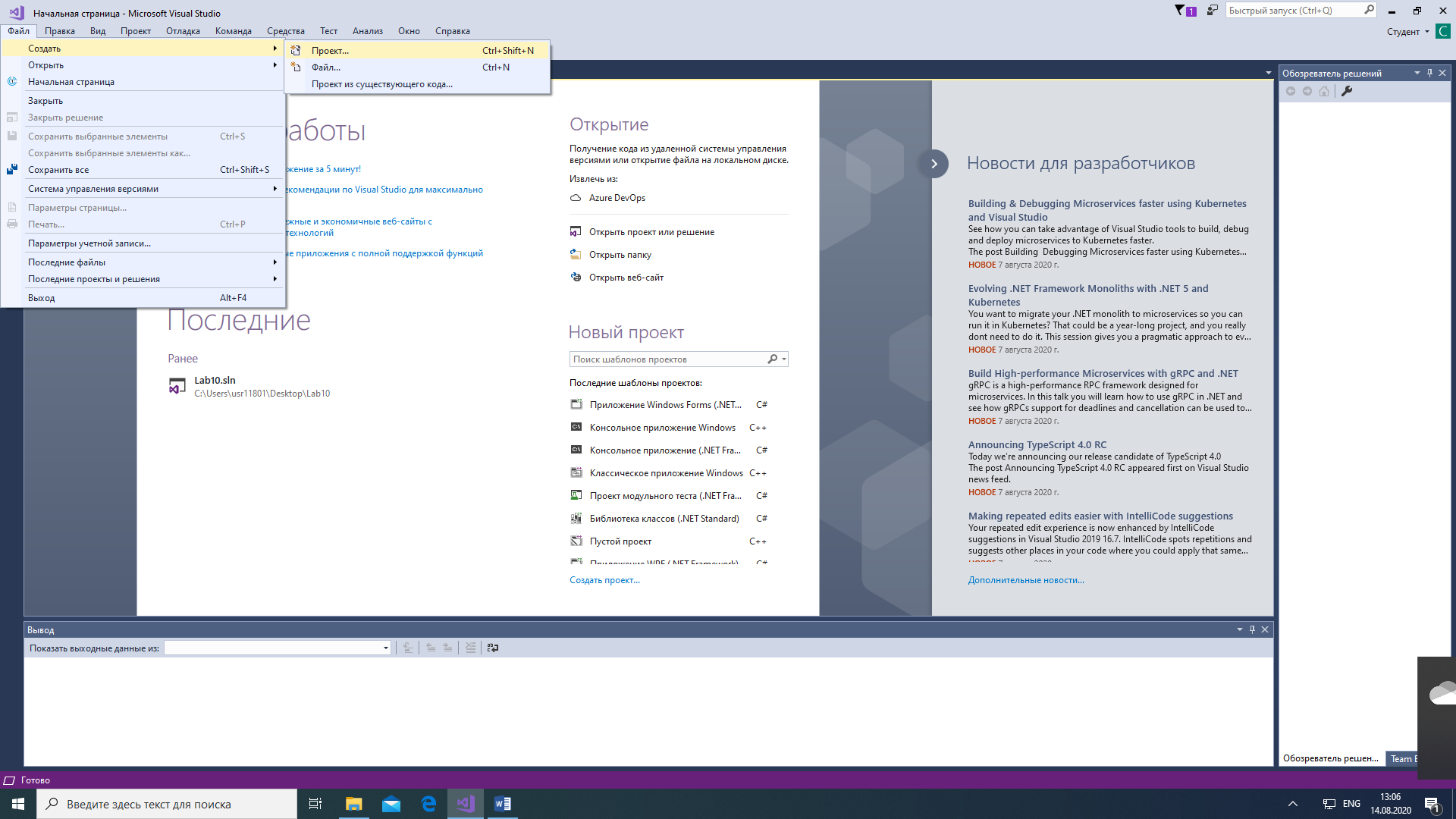
Для запуска среды программирования выберите в меню кнопки Пуск пункт *Visual Studio 2017*. Интерфейс открывшейся страницы представлен на рис. 1.

|  |
| --- |
|  |
| *Рис. 1. Начальная страница приложения Visual Studio 2017* |

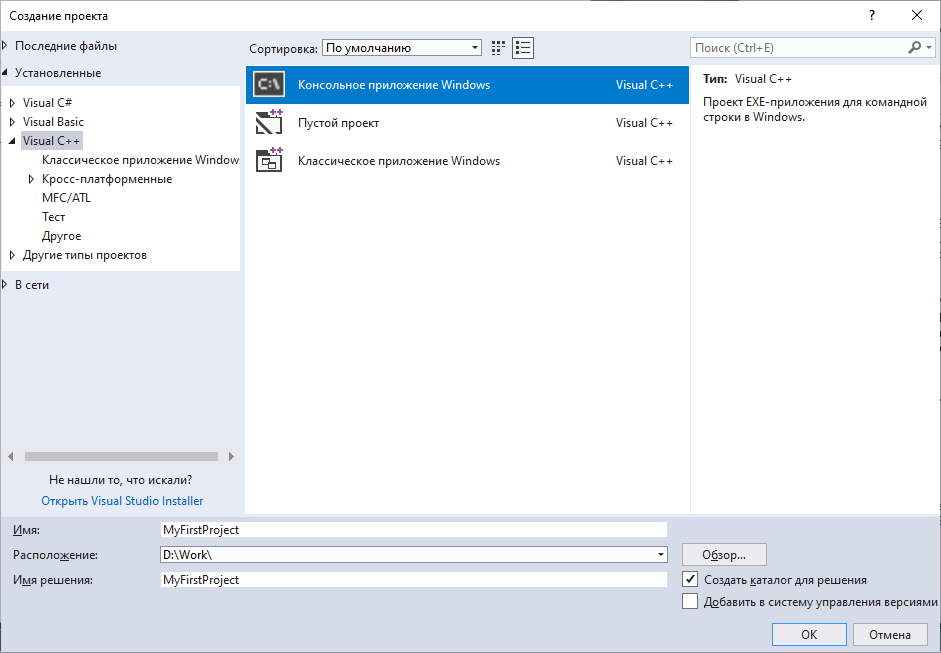
Перед написанием программы необходимо создать проект, который будет содержать все файлы разрабатываемого приложения.

После запуска среды программирования для создания проекта необходимо выбрать **Меню – Файл – Создать – Проект** (рис. 2) или кликнуть на ссылку **Создать проект…** на начальной странице.

В появившемся окне «Создание проекта» (рис. 3) необходимо сначала выбрать язык **Visual C++**, а затем – шаблон создаваемого приложения **Консольное приложение Windows**. Внизу окна нужно указать с помощью кнопки «Обзор» расположение проекта и ввести его имя в соответствующем поле. Имя решения автоматически дублирует имя проекта.



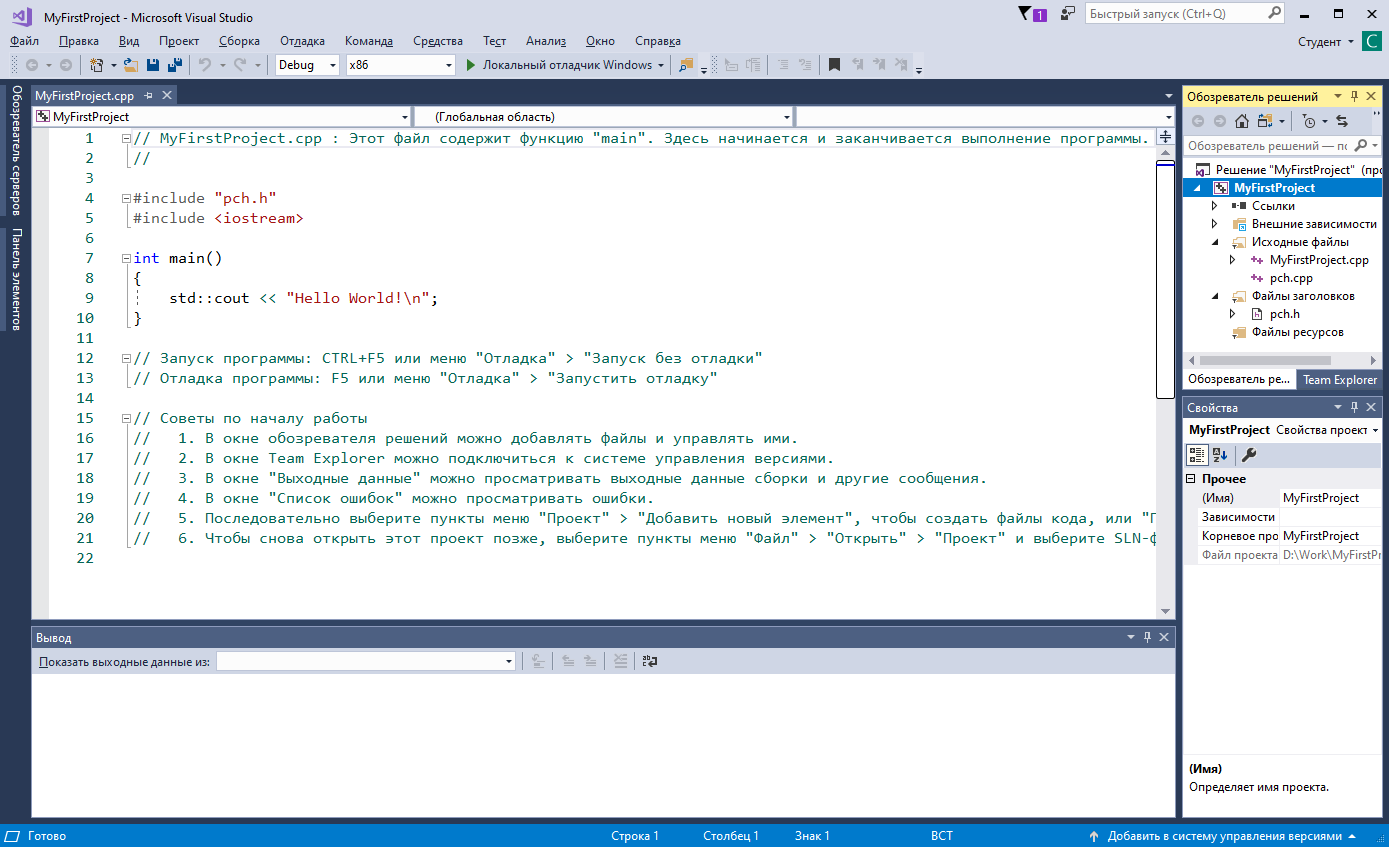
*Рис. 2. Создание проекта*



*Рис. 3. Создание консольного проекта*

Если установлен флажок **Создать каталог для решения**, то все файлы проекта будут находиться в папке с названием проекта.

После нажатия кнопки **ОК** Visual Studio создаст проект (рис. 4).



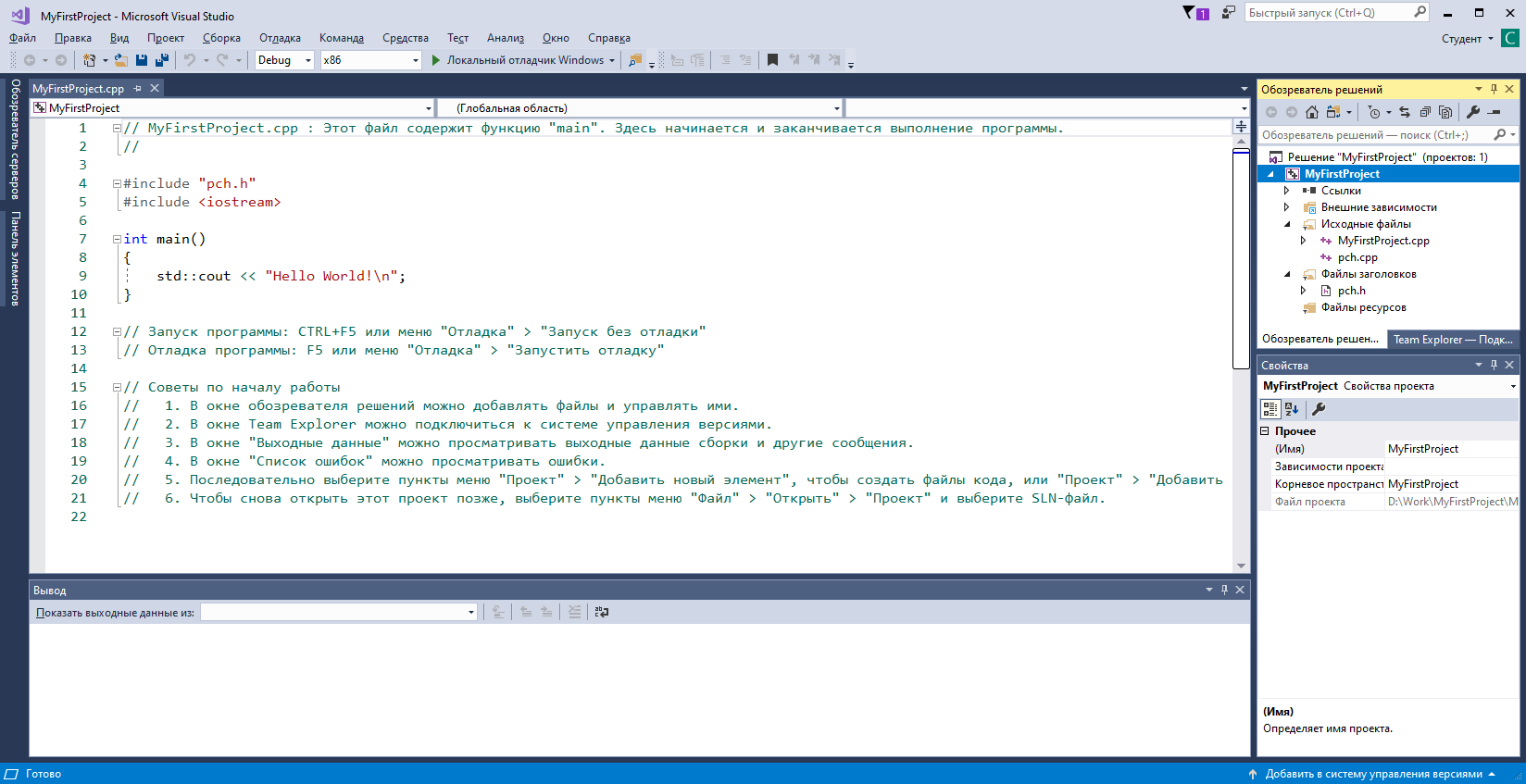
*Рис. 4. Созданный проект*

При таком способе создания проект, кроме файла самой программы (в примере **MyFirstProject.cpp**), содержит файлы предкомпилированного заголовка (**pchx.h**).

Содержимое всех файлов проекта можно просмотреть в окне редактора кода. По умолчанию, при создании нового проекта открывается его главный файл, содержащий функцию **main()**(рис. 4).

**2. Файлы проекта консольного приложения**

С правой стороны находится панель **Обозреватель решений** (5).



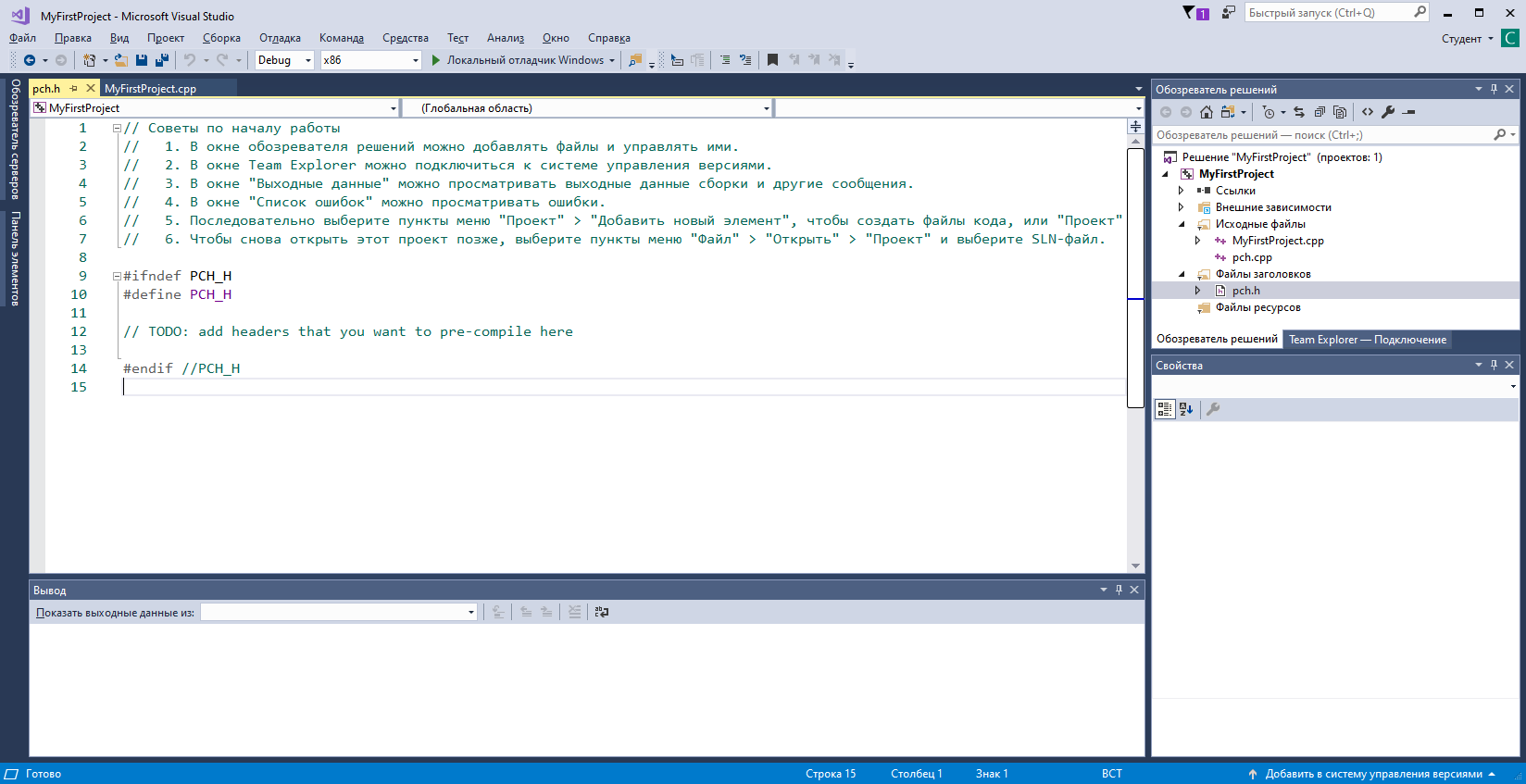
*Рис. 5. Окно «Обозреватель решений»*

Наиболее важными из созданных файлов являются:

– файл исходного кода **<**имя\_проекта**>.cpp**. В этом файле будет находиться основной код программы, в том числе функция **main()**. Имя этого файла должно совпадать с именем проекта.

– заголовочный файл **pch.h**. Содержимое этого файла приведено на рис. 6.

***Упражнение 2.*** Просмотрите содержимое заголовочного файла, которое должно совпадать с приведенным на рис. 6.



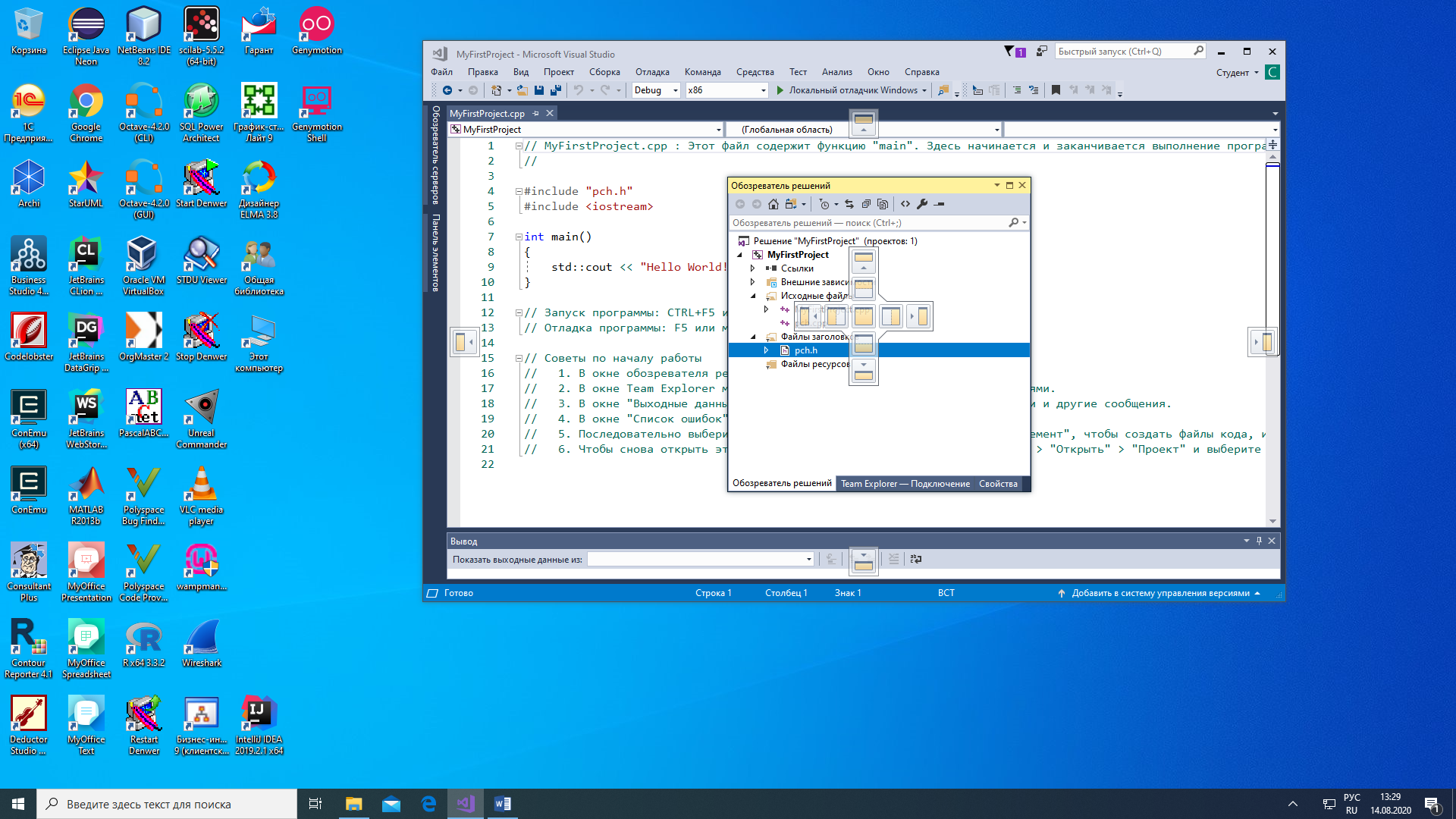
*Рис.6. Файл предварительно скомпилированного заголовка*

Предкомпилированные заголовки (англ. pre-compiled headers) – это метод, позволяющий некоторым компиляторам языков программирования C/C++ уменьшить время компиляции, путем предварительной компиляции редко меняющихся файлов исходного кода приложения. То есть данные файлы компилируются один раз и перекомпилируются только в случае изменения одного из них. Таковыми, как правило, являются заголовочные файлы.

В файлах предкомпилированных заголовков обычно размещаются стандартные заголовки, например, наиболее часто используемые **stdio.h**, **stdlib.h**, **string.h** и **math.h**.

**3. Настройка среды программирования**

В Visual Studio 2017 все окна и панели свободно перемещаются и могут быть либо закрепленными по одной из сторон экрана, либо «висячими». Изменить положение окна или панели можно путем выделения его заголовка и перетаскиванием мышью. При этом на экране появляется разметка, отображающая возможное размещение данного элемента (рис. 7).



*Рис. 7. Перемещение окон*

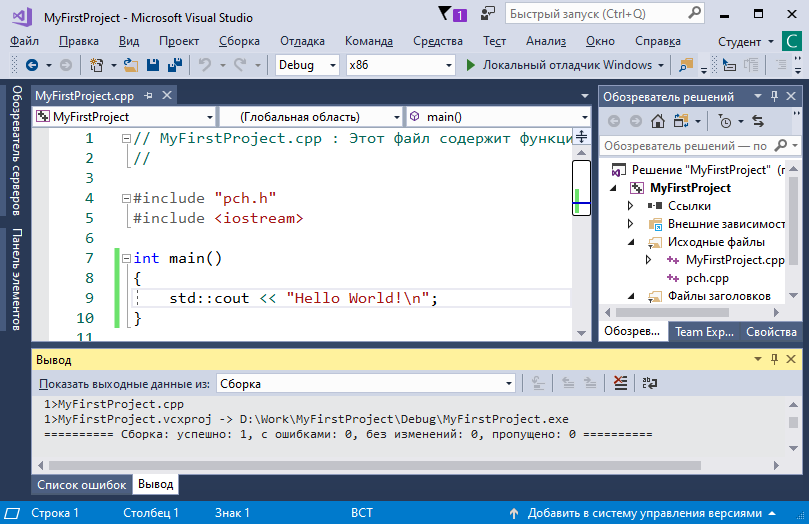
Полный список окон, которые могут отображаться средой программирования, приведен в разделе **Меню – Вид.**

Для удобства отладки программы можно добавить окно **Список ошибок** (**Меню – Вид – Список ошибок**), в котором будут отображаться номера строк кода, содержащие ошибку, и информация о ней.

***Упражнение 3.*** Расположите окна среды программирования наиболее удобным для Вас способом.

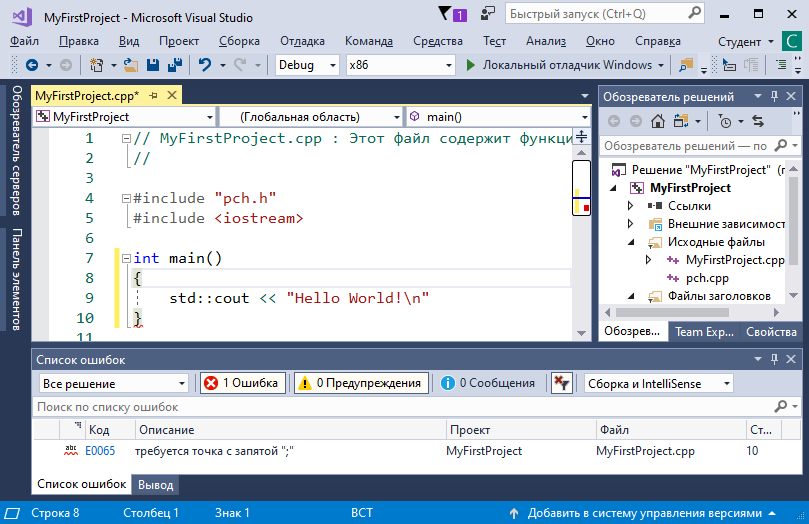
**4. Запуск и отладка программы**

После написания кода программы необходимо выполнить его компиляцию (**Меню – Сборка – Компилировать** или **Ctrl+F5**). Результаты успешной компиляции отображаются в окне **Вывод** (Рис. 8).



*Рис. 8. Сообщение об успешной компиляции*

Если исходный код содержит синтаксические ошибки, то информация о них будет приведена в окне **Список ошибок** (Рис. 9).



*Рис. 9. Сообщение об успешной компиляции*

***Упражнение 4.*** Скопируйте приведенный ниже пример в свой проект. Выполните запуск приложения и убедитесь в отсутствии ошибок.

#include "pch.h"

#include <iostream>

int main()

{

int x, y;

printf("Input x and y: ");

scanf\_s("%d %d", &x, &y);

printf("%d + %d = %d", x, y, x + y);

return 0;

}

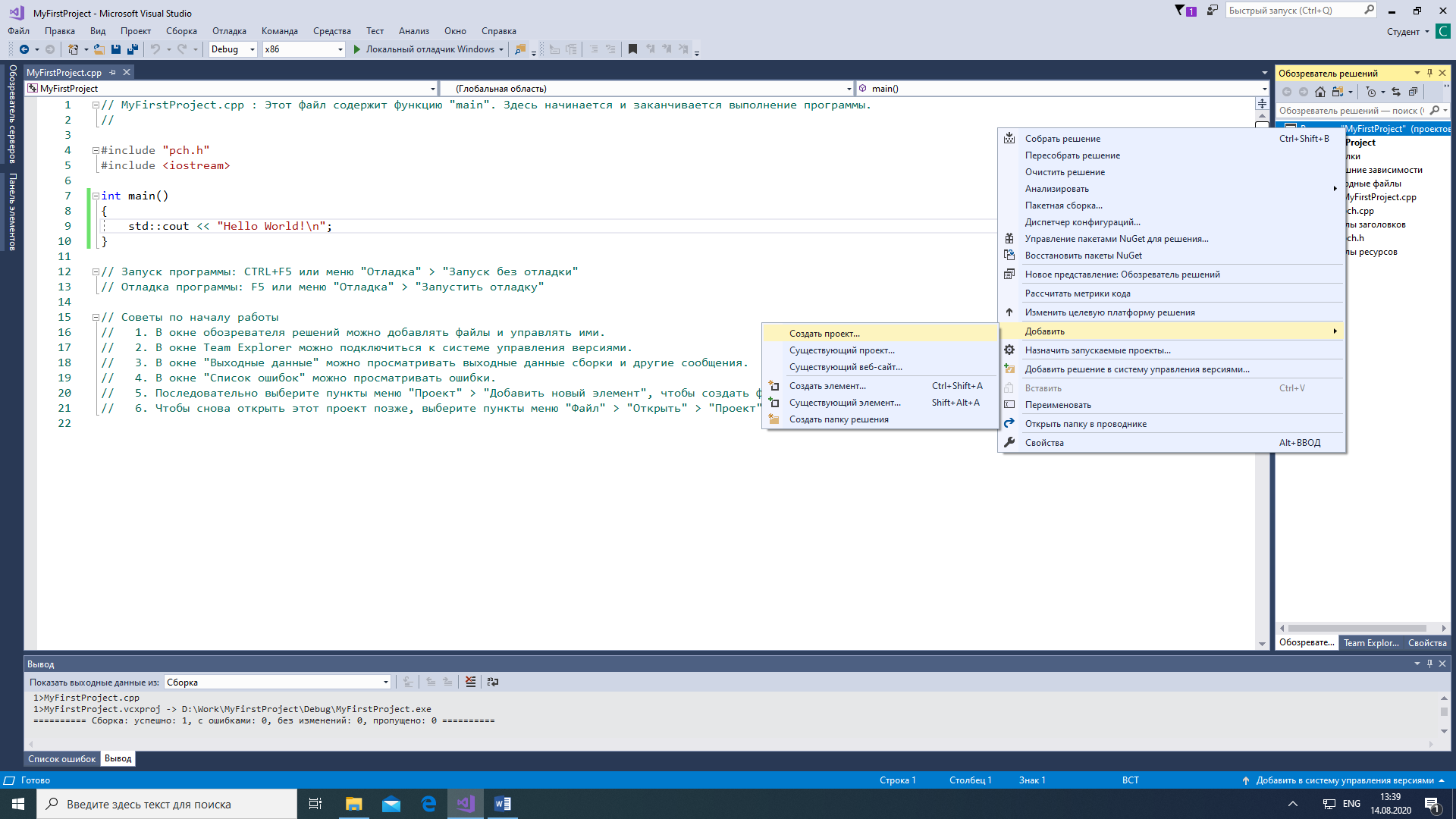
***Упражнение 5.*** Экспериментально определите, какой алгоритм выполняет эта программа. К каждой строке кода напишите комментарий в виде:

int x, y; // объявление переменных

**5. Добавление нового проекта в существующее решение**

Часто для удобства несколько проектов, решающих смежные задачи или части одной большой, объединяют в одно решение. Например, удобно все программы к одной лабораторной работе хранить в одном решении.

Для добавления нового проекта в решение необходимо в окне «Обозреватель решений» выбрать самый верхний уровень, щелкнуть правой кнопкой мыши и выбрать **Добавить – Создать проект…** (рис. 10).



*Рис. 10. Добавление нового проекта в решение*

После этого появится окно для задания настроек нового проекта.

По умолчанию всегда запускается проект, которой создан первым. Чтобы это изменить, нужно в «Обозревателе решений» выбрать другой проект и с помощью контекстного меню его **Назначить запускаемым проектом...**

***Упражнение 6.*** В решение **MyFirstProject** добавьте новый проект с именем **MySecondProject**.

Скопируйте в него текст программы:

#include "pch.h"

#include "stdio.h"

#include "conio.h"

int main() {

int age;

printf("Input your age: ");

scanf\_s("%d", &age);

printf("Hello, human! Your age is %d.\nArtificial Intelligence is immortal!)))", age);

\_getch();

return 0;

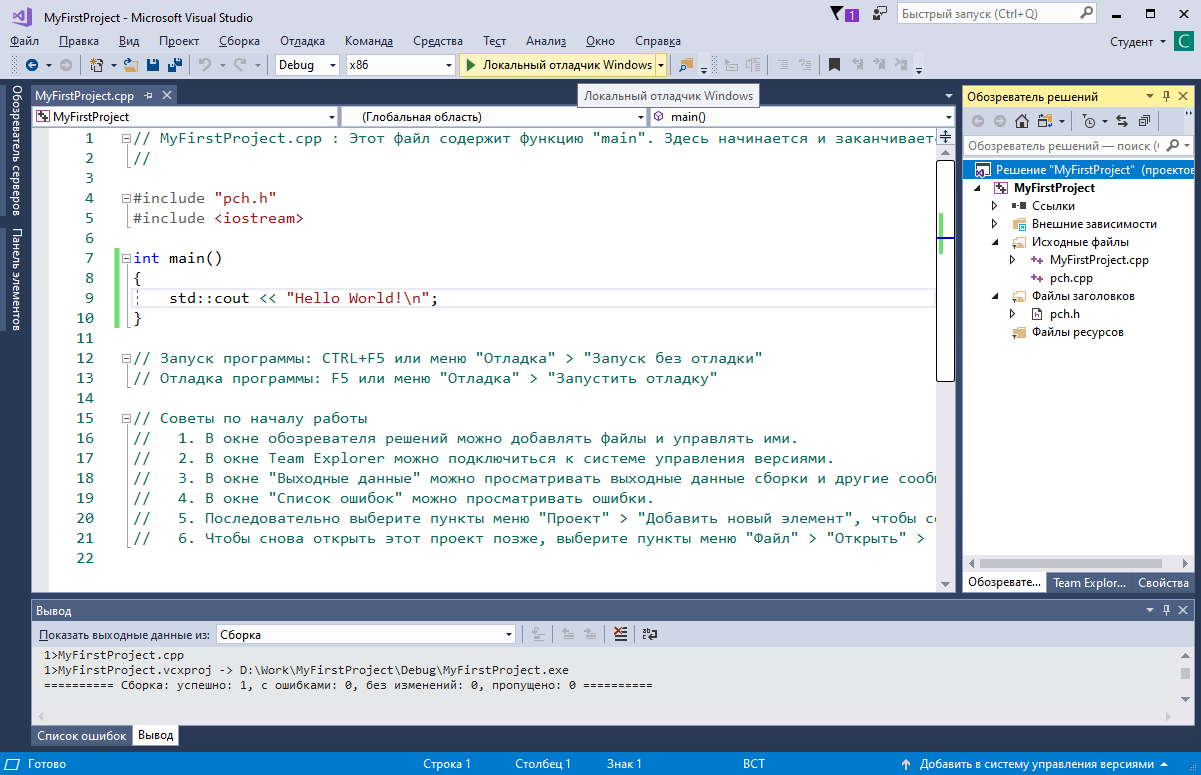
}

Определите, что делает данная программа. Приведите пример входных данных, когда она работает некорректно. Почему?

***Упражнение 7.*** Закомментируйте строку с оператором \_**getch()**. Запустите программу. Что изменилось? Какую роль выполняет этот оператор?

**6. Пошаговое выполнение программы**

Запустить программу на выполнение можно через **Меню – Отладка – Начать отладку**, или кнопкой  на панели инструментов (рис. 11), или нажатием **F5** на клавиатуре.



*Рис. 11. Кнопка Начальный отладчик Windows*

*Отладка* (*Debugging*) – это локализация и устранение ошибок. Эффективным способом отладки является пошаговое выполнение программы строчка за строчкой (*трассировка*).

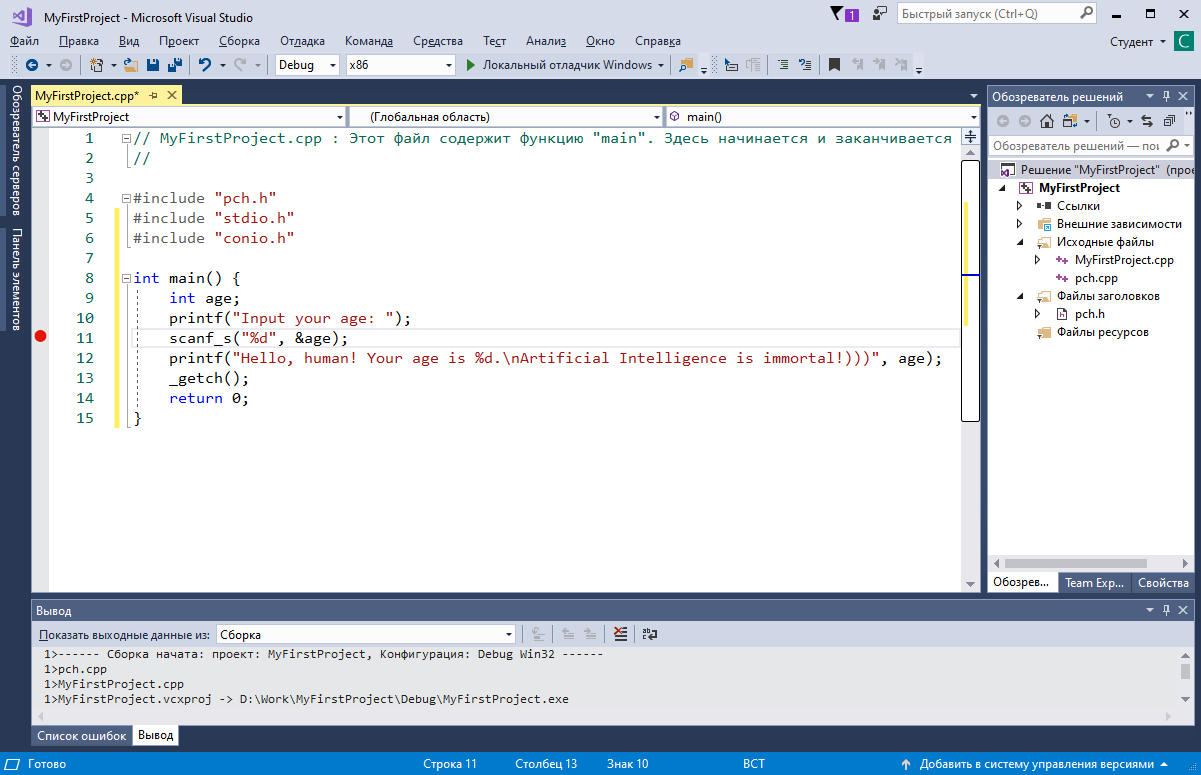
Существует два режима пошагового выполнения программы:

1. трассировка с заходом в тело функции при встрече её вызова в тексте программы (**F11**, );
2. пошаговое выполнение функции как обычной команды без захода в ее тело (**F10**, ).

Иногда нет необходимости просматривать всю программу. В этом случае используют *точки останова* (*BreakPoint*).

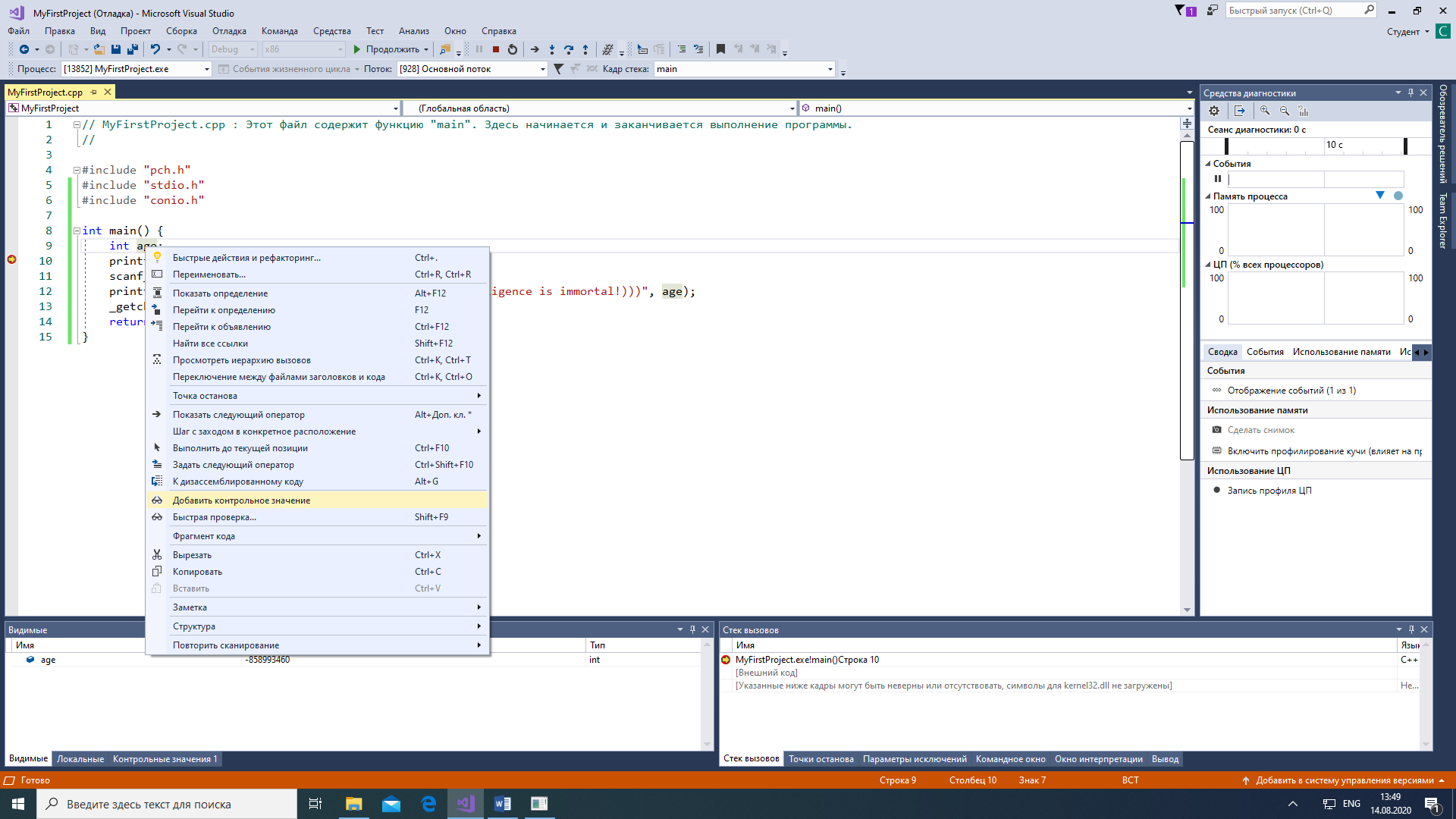
Точка останова позволяет приостановить выполнение программы перед любой выполняемой инструкцией (оператором) для того, чтобы продолжить выполнение программы либо в пошаговом режиме, либо в непрерывном режиме до следующей точки останова.

Чтобы задать точку останова перед некоторым оператором, необходимо установить перед ними текстовый курсор и нажать клавиши **F9** или воспользоваться опцией подменю **Отладка – Создать точку останова** (рис. 12). Рядом со строкой, на которую установлено прерывание, появляется большая красная точка (рис. 12). Убрать точку останова можно повторным нажатием клавиш **F9**, при этом курсор должен находиться на строке, с которой снимается точка прерывания. Также установить и удалить точку прерывания можно кликнув левой кнопкой мыши по месту, где она должна появиться. В программе может быть несколько точек останова.



*Рис. 12. Точка останова*

Для наблюдения за изменением значений переменных в ходе выполнения программы используется окно **Контрольные значения (Меню – Отладка – Окна – Контрольные значения)**. Чтобы добавить переменные в это окно, необходимо во время выполнения выделить требуемую переменную и в контекстном меню выбрать пункт **Добавить контрольное значение** (рис. 13).



*Рис. 13. Добавление переменных в окно Контрольные значения*

Также контрольное значение можно добавить, вписав имя соответствующей переменной в пустую строку окна **Контрольные значения**.

Указанная переменная размещается в окне **Контрольные значения**, создаваемом в нижней части экрана, и, в процессе отладки, через это окно можно наблюдать за изменением размещенных в нем переменных. Удалить переменную из окна можно при помощи клавиши **Del**, предварительно выделив её подсветкой.

***Упражнение 8.*** Выполните программу из упражнения 7 пошагово с просмотром промежуточных значений переменных.

Установите точку останова на строке **scanf\_s("%d", &age);** и запустите программу. Что произойдет?

Посмотрите значение переменной **age**.

Продолжите выполнять программу пошагово, отслеживая изменения переменной **age**.

***Упражнение 9***. Внесите в программу ошибку и посмотрите сообщение в окне ошибок, после чего исправьте внесенную ошибку:

* + удалите символ **;** в конце любой строки;
  + добавьте в код лишний символ;
  + закомментируйте строку **int age;**
  + закомментируйте подключение заголовочного файла **stdafx.h**;
  + в файле **stdafx.h** закомментируйте подключение одной из библиотек;
  + закомментируйте строку **scanf("%d",&age);**
  + удалите символ & в строке **scanf("%d",&age);**
  + закомментируйте строку **return 0;**
  + добавьте лишнюю закрывающую фигурную скобку;
  + удалите закрывающую фигурную скобку.

***Задание для самостоятельного выполнения***

Скопируйте на свой компьютер папку Task4Lab1 c сетевого диска \\lib16\Students\.

Откройте в Visual Studio решение Task4Lab1.sln.

Определите, какие алгоритмы выполняют проекты Task1 и Task2.

Напишите комментарии ко всем строкам кода.