Дисциплина: **Визуальное программирование**

специальность: **информатики**

курс, группа: **1 к 1,2 г**

форма обучения: **заочная**

период проведения занятий: **5 ноября 2020 г.**

вид занятий, кол-во часов: Лекции, 4 часа

преподаватель: **Астахов В.К.**

электронная почта преподавателя: **vadast@mail.ru**

**Литература:**

## 1. Основная литература:

1. Букунов, С. В. Основы объектно-ориентированного программирования: учебное пособие / С. В. Букунов, О. В. Букунова. — СПб.: Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2017. — 196 c. — ISBN 978-5-9227-0713-8. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/74339.html (дата обращения: 11.11.2019). — Режим доступа: для авторизир. пользователей (гриф)
2. Лебедева, Т. Н. Теория и практика объектно-ориентированного программирования: учебное пособие / Т. Н. Лебедева. — 2-е изд. — Челябинск, Саратов: Южно-Уральский институт управления и экономики, Ай Пи Эр Медиа, 2019. — 221 c. — ISBN 978-5-4486-0663-2. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/81498.html (дата обращения: 11.11.2019). — Режим доступа: для авторизир. пользователей
3. Мейер, Б. Объектно-ориентированное программирование и программная инженерия / Б. Мейер. — 3-е изд. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Эр Медиа, 2019. — 285 c. — ISBN 978-5-4486-0513-0. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/79706.html (дата обращения: 11.11.2019). — Режим доступа: для авторизир. пользователей

## 2. Дополнительная литература:

1. Борисенко, В. В. Основы программирования / В. В. Борисенко. — М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 323 c. — ISBN 978-5-9556-00039-0. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/52206.html (дата обращения: 11.11.2019). — Режим доступа: для авторизир. пользователе
2. Мухаметзянов, Р. Р. Основы программирования в Delphi: учебно-методическое пособие / Р. Р. Мухаметзянов. — Набережные Челны: Набережночелнинский государственный педагогический университет, 2017. — 137 c. — ISBN 2227-8397. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/66811.html (дата обращения: 11.11.2019). — Режим доступа: для авторизир. пользователей
3. Разработка Windows-приложений в среде программирования Visual Studio.Net : учебно-методическое пособие по дисциплине Информатика и программирование / составители Ю. А. Воронцов, А. Г. Ерохин. — М.: Московский технический университет связи и информатики, 2016. — 20 c. — ISBN 2227-8397. — Текст: электронный // Электронно-библиотечная система IPR BOOKS: [сайт]. — URL: http://www.iprbookshop.ru/61536.html (дата обращения: 11.11.2019). — Режим доступа: для авторизир. пользователей

**Интернет- ресурсы**

1. Бесплатные видеолекции: Визуальное программирование... [Электронный ресурс]: офиц.сайт. — Электрон. дан. — Режим доступа: http://www. edumarket.ru/t raining/design/ graphics/74481
2. Библиотека программиста Видео-лекции. Язык программирования Си… [Электронный ресурс]: офиц.сайт. — Электрон. дан. — Режим доступа: http://www.vk.com/page-54530371\_46988280
3. Видеолекция по визуальному программированию ... [Электронный ресурс]: офиц.сайт. — Электрон. дан. — Режим доступа: http://www. ezolife.info/?p=13453
4. Знакомство со средой Microsoft Visual Studio 2010 и настройка компилятора языка С. Стандартный ввод-вывод [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа <https://intuit.ru/studies/courses/627/483/lecture/10944>
5. Научная электронная библиотека eLIBRARY.ru [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа: http://www.elibrary.ru/ defaultx.asp
6. Научная электронная библиотека IPRbooks.ru [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа: http://www. iprbooks.ru (по паролю)
7. Научная электронная онлайн-библиотека Порталус [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа: http://www. portalus.ru
8. Программирование. Онлайн видео уроки «Уроки программирования на Delphi. [Электронный ресурс]: офиц.сайт. — Электрон. дан. — Режим доступа: http://www. compteacher.ru
9. Среда разработки Visual Studio [Электронный ресурс]: раздел Информационные технологии. — Электрон. дан. — Режим доступа <https://professorweb.ru/my/programs/visual-studio/level1/1_2.php>

**Задания по темам и датам**

| **дата, время**  **занятия** | **тема**  **занятия** | **кол-во часов** | **вопросы для изучения и обсуждения** | **литература** | **контрольные вопросы, задания** |
| --- | --- | --- | --- | --- | --- |
| 05.11.2020  13.00-14.30  офф-лайн | Лекция №1. | 2 | 1.Формы Windows и пользовательский интерфейс  2.Основы работы с интегрированной средой разработки Microsoft Visual Studio | 1.Осн. и доп. литература, интернет-ресурсы | 1.Изучить представленный ниже материал и рекомендованную литературу.  (присылать материалы преподавателю для проверки НЕ НУЖНО) |
| 05.11.2020  14.35-16.05  офф-лайн | Лекция №2. | 2 | 1. Обзор контейнерных элементов управления.  2. Управляющие элементы, обеспечивающие выбор из фиксированного набора альтернатив, предоставленных пользователю | 1.Осн. и доп. литература, интернет-ресурсы | 1.Изучить представленный ниже материал и рекомендованную литературу.  (присылать материалы преподавателю для проверки НЕ НУЖНО) |

Экзамен (будет зимой)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  | Перечень вопросов к экзамену по дисциплине «Визуальное программирование»  (60 вопросов к экзамену) | 1.Осн. и доп. литература, интернет-ресурсы | Перечень вопросов к экзамену по дисциплине Визуальное программирование находится на сайте во вкладке: Студенту.  Далее: Вопросы к зачетам и экзаменам-Прикладная информатика (бакалавриат)-Полный перечень вопросов и заданий (Cкачать. ZIP).  Затем в скачанном списке ищем дисциплину «Визуальное программирование» и открываем файл в word. |

Краткий материал по дисциплине **Визуальное программирование** для консультаций на 05.11.2020 г.

***Лекция №1***

<https://intuit.ru/studies/courses/627/483/lecture/10944>

**Знакомство со средой Microsoft Visual Studio 2010 и настройка компилятора языка С. Стандартный ввод-вывод**

Язык **С** (читается как *Си*) в основе своей был создан в 1972 г. как язык для операционной системы *UNIX* [[1.2](https://intuit.ru/studies/courses/627/483/literature#literature.1.2)]. Автором этого языка считается Денис М. Ритчи (DENNIS M. RITCHIE).

Популярность языка **С** обусловлена, прежде всего тем, что большинство операционных систем были написаны на языке С. Его начальное распространение было задержано из-за того, что не было удачных компиляторов.

Несколько лет не было единой политики в стандартизации языка **С**. В начале 1980-х г. в Американском национальном институте стандартов (*ANSI*) началась работа *по* стандартизации языка **С**. В 1989 г. работа комитета *по* языку С была ратифицирована, и в 1990 г. был издан первый официальный документ *по* стандарту языка **С**. Появился стандарт 1989, т.е. **С89** [[1.3](https://intuit.ru/studies/courses/627/483/literature#literature.1.3)].

К разработке стандарта *по* языку **С** была также привлечена Международная организация *по* стандартизации (*ISO*). Появился стандарт *ISO*/IEC 9899:1990, или *ANSI* C99 языка **С** [[1.3](https://intuit.ru/studies/courses/627/483/literature#literature.1.3)].

В данном пособии за основу принимается *стандарт языка* **С** от 1989 г. и написание программ будет выполняться в среде разработки *Visual Studio* 2010.

Язык **С** является прежде всего языком высокого уровня, но в нем заложены возможности, которые позволяют программисту (пользователю) работать непосредственно с аппаратными средствами компьютера и общаться с ним на достаточно низком уровне [[1.3](https://intuit.ru/studies/courses/627/483/literature#literature.1.3)]. Многие *операции*, выполняемые на языке **С**, сродни языку Ассемблера. Поэтому язык С часто называют языком среднего уровня.

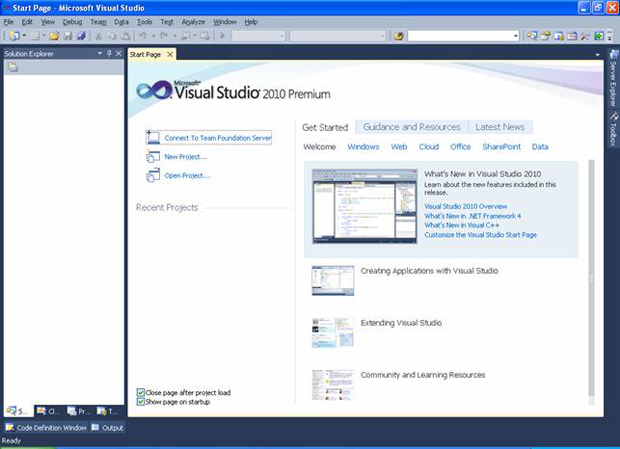
Для написания программ в практических разделах данного учебного пособия будет использоваться *компилятор* языка **С++**, а *программирование* будет вестись в среде **Microsoft Visual Studio 2010**. Предполагается, что на компьютере установлена эта интегрированная среда.

**Microsoft Visual Studio 2010** доступна в следующих вариантах:

* **Express** – бесплатная среда разработки, включающая только базовый набор возможностей и библиотек.
* **Professional** – поставка, ориентированная на профессиональное создание программного обеспечения, и командную разработку, при которой созданием программы одновременно занимаются несколько человек.
* **Premium** – издание, включающее дополнительные инструменты для работы с исходным кодом программ и создания баз данных.
* **Ultimate** – наиболее полное издание Visual Studio, включающие все доступные инструменты для написания, тестирования, отладки и анализа программ, а также дополнительные инструменты для работы с базами данных и проектирования архитектуры ПО.

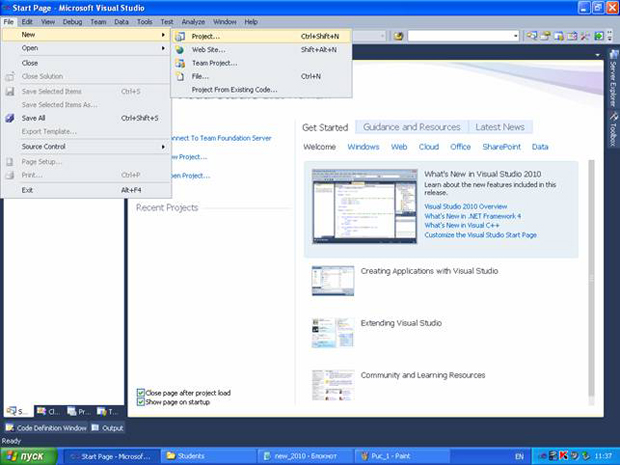
Отличительной особенностью среды **Microsoft Visual Studio 2010** является то, что она поддерживает работу с несколькими языками программирования и программными платформами. Поэтому, перед тем, как начать создание программы на языке С, необходимо выполнить несколько подготовительных шагов *по* созданию проекта и выбора и настройки компилятора языка С для трансляции исходного кода

После запуска **Microsoft Visual Studio 2010** появляется следующая стартовая страница, которая показана на [рис. 1.1](https://intuit.ru/studies/courses/627/483/lecture/10944?page=1#image.1.1).



**Рис. 1.1.**Стартовая страница Visual Studio 2010

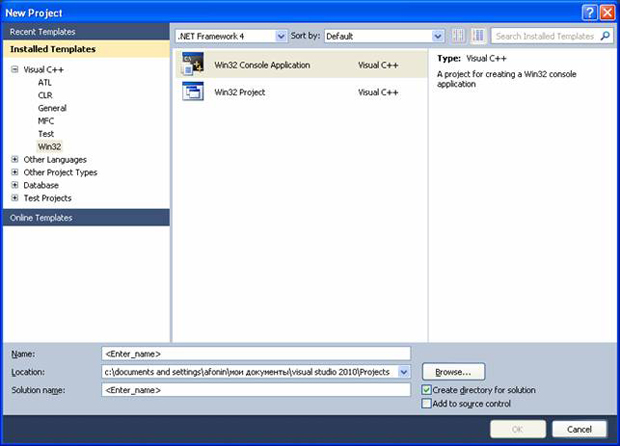
Следующим шагом является создание нового проекта. Для этого в *меню* **File** необходимо выбрать **New – Project** (или комбинацию клавиш **Ctrl + Shift + N** ). Результат выбора пунктов *меню* для создания нового проекта показан на [рис. 1.2](https://intuit.ru/studies/courses/627/483/lecture/10944?page=1#image.1.2).



**Рис. 1.2.**Окно с выбором нового проекта

Среда *Visual Studio* отобразит окно **New Project**, в котором необходимо выбрать тип создаваемого проекта. *Проект* (*project*) используется в *Visual Studio* для логической группировки нескольких файлов, содержащих исходный код, на одном из поддерживаемых языков программирования, а также любых вспомогательных файлов. Обычно после сборки проекта (которая включает компиляцию всех входящих в проект файлов исходного кода) создается один исполняемый *модуль*.

В окне **New Project** следует развернуть узел *Visual* С++, обратиться к пункту Win32 и на центральной панели выбрать Win32 *Console Application*. Выбор этой опции показан на рис. [рис. 1.3](https://intuit.ru/studies/courses/627/483/lecture/10944?page=2#image.1.3).

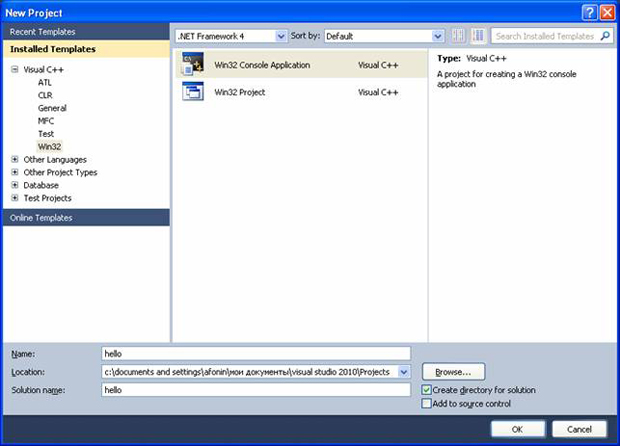


**Рис. 1.3.**Выбор типа проекта

Затем в *поле* редактора **Name** (где *по* умолчанию имеется <Enter\_name>) следует ввести имя проекта, например, **hello**. В *поле* **Location** можно указать *путь* размещения проекта, или выбрать *путь* размещения проекта с помощью клавиши (кнопки) Browse. *По* умолчанию проект сохраняется в специальной папке Projects. Пример выбора имени проекта показано на [рис. 1.4](https://intuit.ru/studies/courses/627/483/lecture/10944?page=2#image.1.4).

Одновременно с созданием проекта *Visual Studio* создает решение. *Решение* (solution) – это способ объединения нескольких проектов для организации более удобной работы с ними.

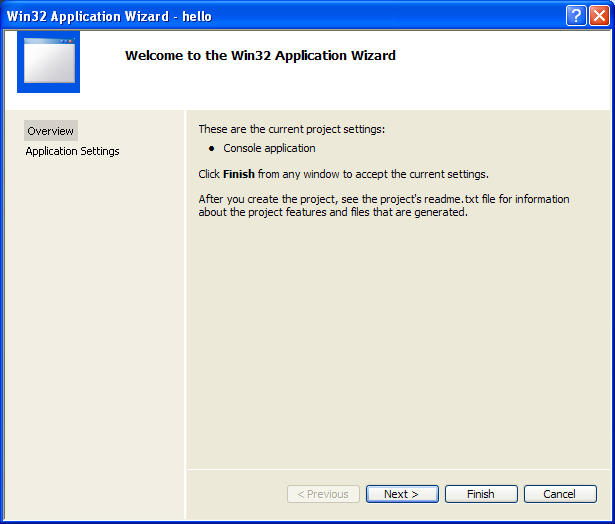
После нажатия кнопки **OK** откроется окно **Win32 Application Wizard** (мастер создания приложений для операционных систем *Windows*), показанное на [рис. 1.5](https://intuit.ru/studies/courses/627/483/lecture/10944?page=2#image.1.5).



**Рис. 1.4.**Пример задания имени проекта

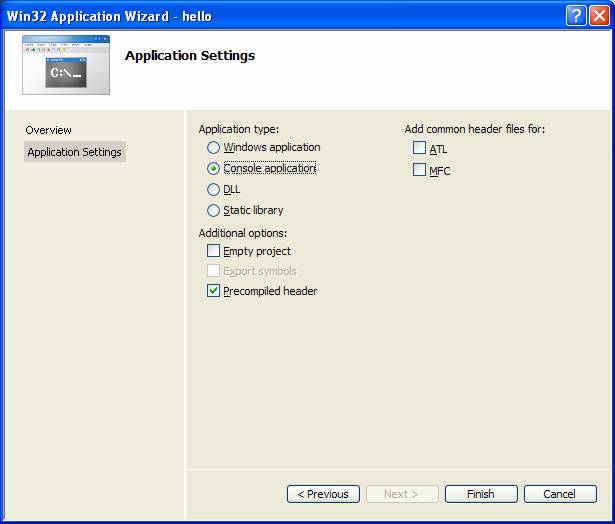
Выбор имени проекта может быть достаточно произвольным: допустимо использовать числовое *значение*, допустимо имя задавать через буквы русского алфавита.

В дальнейшем будем использовать имя, набранное с помощью букв латинского алфавита и, может быть, с добавлением цифр.



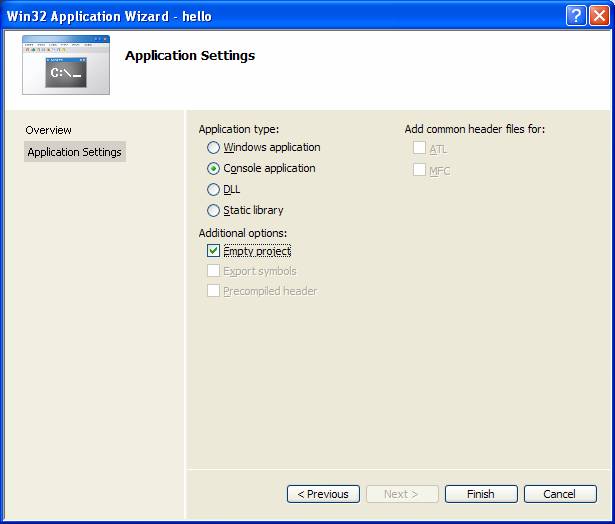
**Рис. 1.5.**Мастер создания приложения

На первой странице представлена *информация* о создаваемом проекте, на второй можно сделать первичные настройки проекта. После обращения к странице **Application Settings**, или после нажатия кнопки **Next** получим окно, показанное на рис. [рис. 1.6](https://intuit.ru/studies/courses/627/483/lecture/10944?page=2#image.1.6).



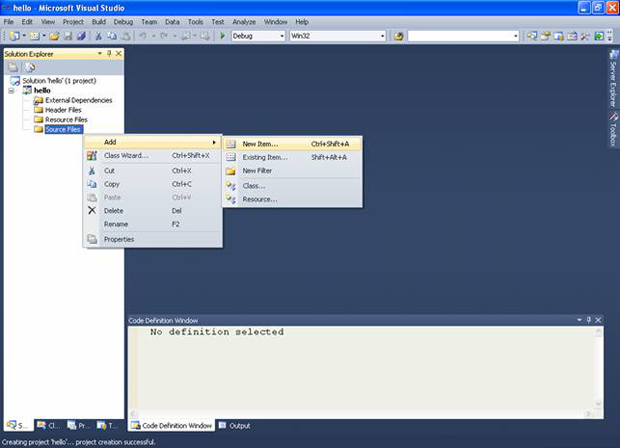
**Рис. 1.6.**Страница мастера настройки проекта по умолчанию

В дополнительных опциях ( **Additional options** ) следует поставить галочку в *поле* **Empty project** (пустой проект) и снять (убрать) галочку в *поле* **Precompiled header**. Получим экранную форму, показанную на [рис. 1.7](https://intuit.ru/studies/courses/627/483/lecture/10944?page=2#image.1.7).



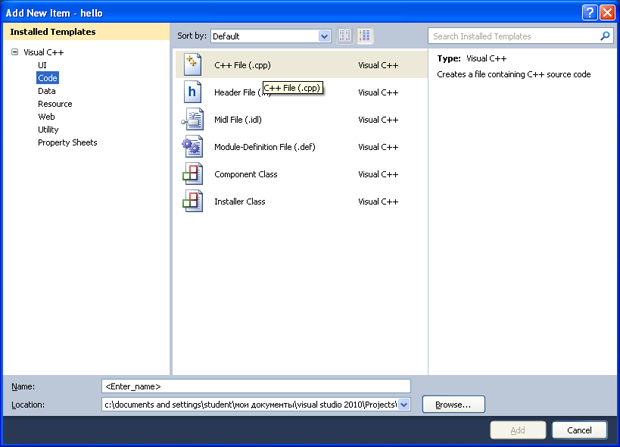
**Рис. 1.7.**Выполненная настройка мастера приложений

Здесь и далее будут создавать проекты *по* приведенной схеме, т.е. проекты в консольном приложении, которые должны создаваться целиком программистом (за счет выбора **Empty project** ). После нажатия кнопки **Finish**, получим экранную форму, показанную на [рис. 1.8](https://intuit.ru/studies/courses/627/483/lecture/10944?page=2#image.1.8), где приведена последовательность действий добавления файла для создания исходного кода к проекту. Стандартный *путь* для этого: подвести *курсор* мыши к папке **Source Files** из узла **hello** в левой части открытого проекта приложения, выбрать **Add** и **New Item** (новый элемент).



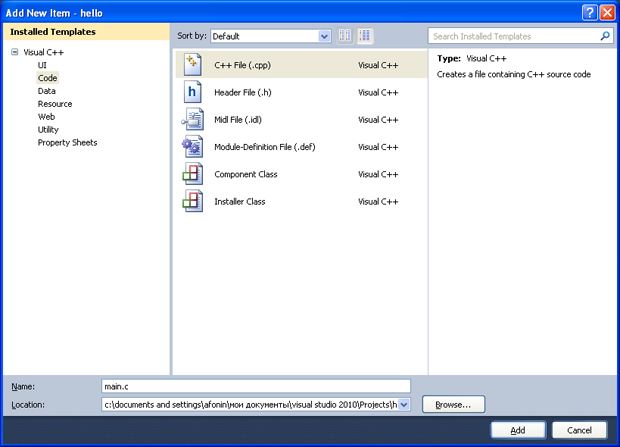
**Рис. 1.8.**Меню добавления нового элемента к проекту

После выбора (нажатия) **New Item** получим окно, показанное на [рис. 1.9](https://intuit.ru/studies/courses/627/483/lecture/10944?page=2#image.1.9), где через *пункт* *меню* **Code** узла **Visual C++** выполнено обращение к центральной части панели, в которой осуществляется выбор типа файлов. В данном случае требуется обратиться к закладке **C++ File** (.cpp).



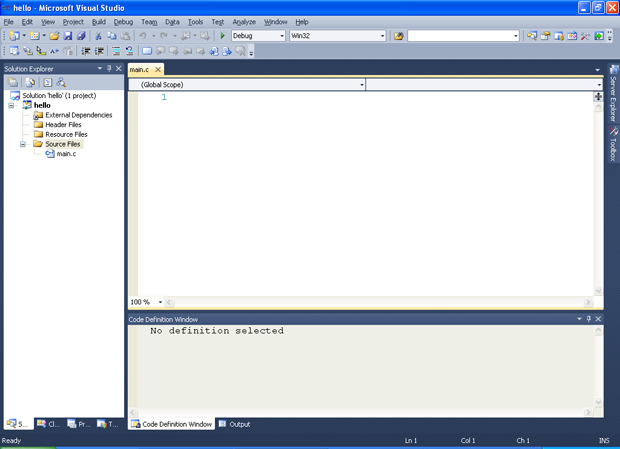
**Рис. 1.9.**Окно выбора типа файла для подключения к проекту

Теперь в *поле* редактора **Name** (в нижней части окна) следует задать имя нового файла и указать расширение **".с"**. Например, **main.c**. *Имя файла* может быть достаточно произвольным, но имеется негласное соглашение, что *имя файла* должно отражать его назначение и логически описывать исходный код, который в нем содержится. В проекте, состоящем из нескольких файлов, имеет смысл выделить *файл*, содержащий главную функцию программы, с которой она начнет выполняться. В данном пособии такому файлу мы будем задавать имя **main.c**, где расширение **.с** указывает на то, что этот *файл* содержит исходный код на языке **С**, и он будет транслироваться соответствующим компилятором. Программам на языке **С** принято давать расширение **.с**. После задания имени файла в *поле* редактора **Name**, получим форму, показанную на [рис. 1.10](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.10).



**Рис. 1.10.**Задание имени файла, подключаемому к проекту

Затем следует нажать кнопку **Add**. Вид среды *Visual Studio* после добавления первого файла к проекту показан на [рис. 1.11](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.11). Добавленный *файл* отображается в дереве **Solution Explorer** под узлом *Source Files* (файлы с исходным кодом), и для него автоматически открывается редактор.



**Рис. 1.11.**Подключение файла проекта

На [рис. 1.11](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.11) в левой панели в папке **Solution Explorer** отображаются файлы, включенные в проект в папках. Приведем описание.

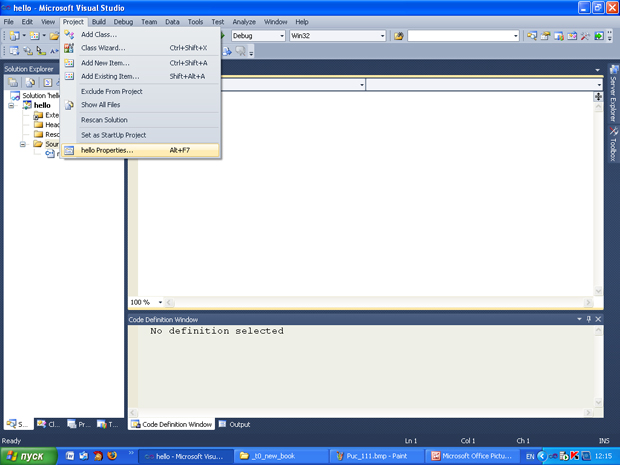
*Папка* **Source Files** предназначена для файлов с исходным кодом. В этой папке отображаются файлы с расширением **.с**.

*Папка* **Header Files** содержит заголовочные файлы с расширением **.h**.

*Папка* **Resource Files** содержит *файлы ресурсов*, например изображения и т. д.

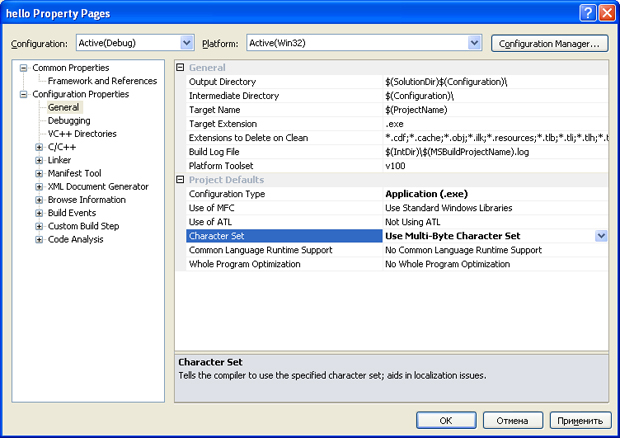
*Папка* **External Dependencies** отображает файлы, не добавленные явно в проект, но использующиеся в файлах исходного кода, например включенные при помощи директивы #include. Обычно в папке **External Dependencies** присутствуют заголовочные файлы стандартной библиотеки, использующиеся в проекте.

Следующий шаг состоит в настройке проекта. Для этого в *меню* **Project** главного *меню* следует выбрать **hello Properties** (или с помощью последовательного нажатия клавиш Alt+F7). Пример обращения к этому пункту *меню* показан на [рис. 1.12](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.12).



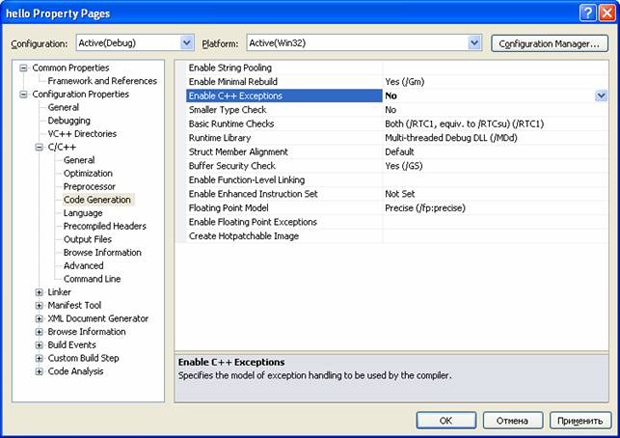
**Рис. 1.12.**Обращение к странице свойств проекта

После того как произойдет открытие окна свойств проекта, следует обратиться (с левой стороны) к **Configuration Properties**. Появится ниспадающий *список*, который показан на [рис. 1.13](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.13). Выполнить обращение к узлу *General*, и через него в правой панели выбрать **Character Set**, где установить свойство **Use Multi-byte Character Set**. Настройка **Character Set** (набор символов) позволяет выбрать, какая *кодировка* символов – *ANSI* или UNICODE – будет использована при компиляции программы. Для совместимости со стандартом C89 мы выбираем **Use Multi-Byte Character Set**. Это позволяет использовать многие привычные функции, например, функции *по* выводу информации на *консоль*.



**Рис. 1.13.**Меню списка свойств проекта

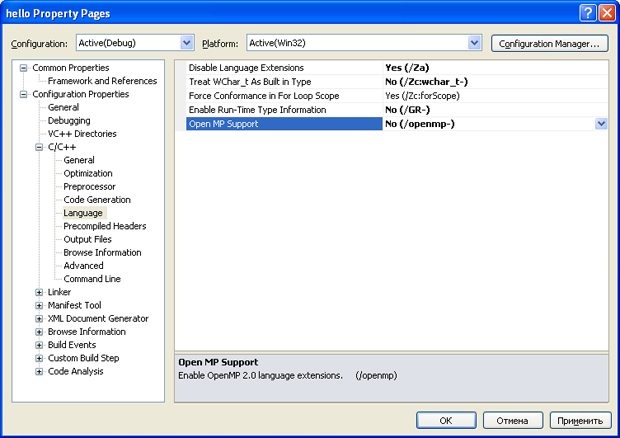
После сделанного выбора, показанного на [рис. 1.13](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.13), следует нажать кнопку Применить. Затем следует выбрать узел **С/С++** и в ниспадающем *меню* выбрать *пункт* **Code Generation**, через который следует обратиться в правой части панели к закладке **Enable C++ Exceptions**, для которой установить **No** (запрещение исключений С++). Результат установки выбранного свойства показан на [рис. 1.14](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.14). После произведенного выбора нажать кнопку **Применить**.



**Рис. 1.14.**Страница свойств для запрещения исключений С++

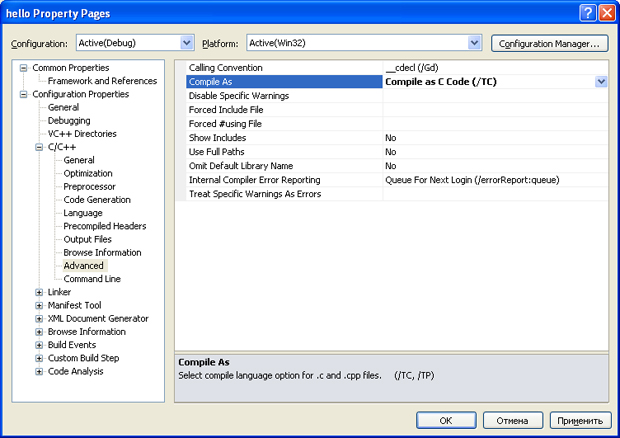
Далее в ниспадающем *меню* узла **С/С++** необходимо выбрать *пункт* **Language** и через него обратиться в правую часть панели, где установить следующие свойства: свойство **Disable Language Extensions** (дополнительные языковые расширения фирмы Microsoft) в **Yes (/Za)**, свойство **Treat wchar\_t as Built-in Type** (рассматривать тип wchar\_t как встроенный тип) установить в **No (/Zc:wchar\_t–)**, свойство **Force Conformance in For Loop Scope** (соответствие стандарту определения локальных переменных в операторе *цикла* for ) установить в Yes(/Zc:forScope), свойство **Enable Run-Time Type Info** (разрешить информацию о типах во *время выполнения*) установить в **No (/GR–)**, свойство **Open MP Support** (разрешить расширение *Open* MP – используется при написании программ для многопроцессорных систем) установить в **No(/openmp–)**.

Результат выполнения этих действий показан на [рис. 1.15](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.15).



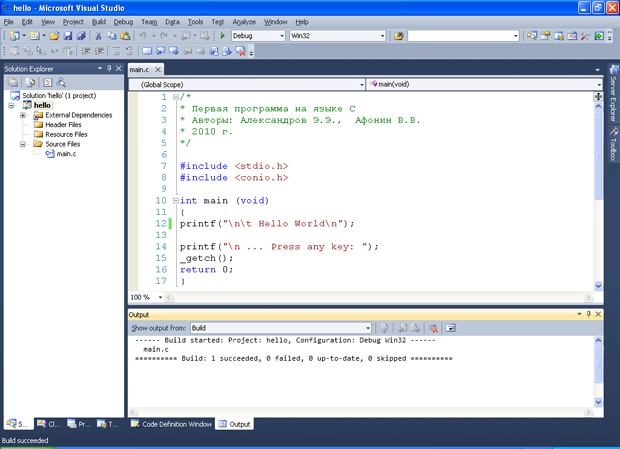
**Рис. 1.15.**Страница свойств закладки Language

После выполнения указанных действий следует нажать клавишу **Применить**. Далее в ниспадающем списке узла **С/С++** следует выбрать *пункт* **Advanced** и в правой панели изменить свойство **Compile As** в свойство компиляции языка **С**, т.е. **Compile as C Code (/TC)**. Результат установки компилятора языка **С** показан на [рис. 1.16](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.16).



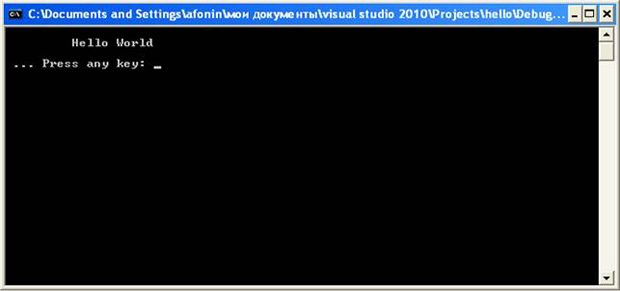
**Рис. 1.16.**Результат выбора режима компиляции языка С

После нажатия клавиш **Применить** и **ОК** сначала откроется подготовленный проект с пустым полем редактора кода, в котором можно начать писать программы. В этом редакторе наберем программу, выводящую традиционное приветствие "Hello World". Для компиляции созданной программы можно обратиться в *меню* **Build**, или, например, набрать клавиши *Ctr*+F7. В случае успешной компиляции получим следующую экранную форму, показанную на [рис. 1.17](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.17).



**Рис. 1.17.**Успешно откомпилированная первая программа на языке С

Для приведенного кода программы *запуск* на ее *исполнение* из окна редактора в *Visual Studio* 2010 можно нажать клавишу **F5**. [рис. 1.18](https://intuit.ru/studies/courses/627/483/lecture/10944?page=3#image.1.18) показан результат исполнения первой программы.



**Рис. 1.18.**Консольный вывод первой программы на языке С

Произведем разбор первой программы. Во-первых, надо отметить, что в языке **С** нет стандартных инструкций (операторов) для вывода сообщений на *консоль* (окно пользователя). В языке **С** предусматриваются специальные библиотечные файлы, в которых имеются функции для этих целей. В приведенной программе используется заголовочный *файл* с именем stdio.h (*стандартный ввод*–*вывод*), который должен быть включен в начало программы. Для вывода сообщения на *консоль* используется *функция* printf(). Для работы с консолью включен также заголовочный *файл* conio.h., который поддерживает функцию \_getch(), которая извлекает символ из потока ввода, т. е. она предназначена для приема сообщения о нажатии какой-либо (почти любой) клавиши на клавиатуре. С другими компиляторами, возможно, потребуется getch(), т.е. без префиксного нижнего подчеркивания. Строка программы

int main (void)

сообщает системе, что именем программы является main() – главная *функция*, и что она возвращает *целое число*, о чем указывает аббревиатура **"int"**. Имя main() – это специальное имя, которое указывает, где *программа* должна начать выполнение [[1.1](https://intuit.ru/studies/courses/627/483/literature#literature.1.1)]. Наличие круглых скобок после слова main() свидетельствует о том, что это *имя функции*. Если содержимое круглых скобок отсутствует или в них содержится *служебное слово* **void**, то это означает, что в функцию main() не передается никаких аргументов. *Тело функции* main() ограничено парой фигурных скобок. Все утверждения программы, заключенные в *фигурные скобки*, будут относиться к функции main().

В теле функции main() имеются еще три функции. Во-первых, функции printf() находятся в библиотеке компилятора языка **С**, и они печатают или отображают те аргументы, которые были подставлены вместо параметров. Символ "\n" составляет единый символ *newline* (новая строка), т.е. с помощью этого символа осуществляется перевод на новую строку. Символ "\t" осуществляет табуляцию, т.е. начало вывода результатов программы с отступом вправо.

*Функция* без параметров \_getch() извлекает символ из потока ввода (т.е. ожидает нажатия почти любой клавиши). С другими компиляторами, возможно, потребуется getch(), т.е. без префиксного нижнего подчеркивания.

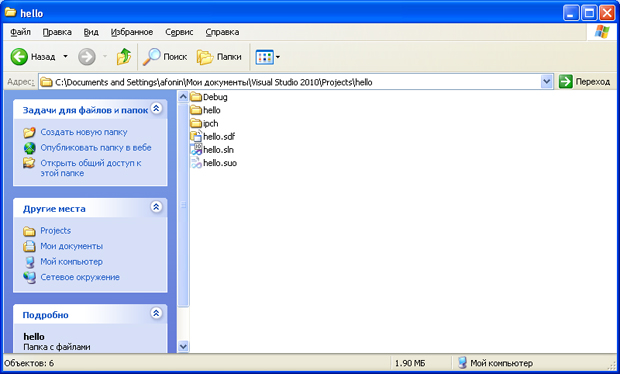
Последнее утверждение в первой программе

return 0;

указывает на то, что выполнение функции main() закончено и что в систему возвращается *значение* 0 (*целое число*). Нуль используется в соответствии с соглашением об индикации успешного завершения программы [[1.3](https://intuit.ru/studies/courses/627/483/literature#literature.1.3)].

В завершение следует отметить, что все действия в программе завершаются символом точки с запятой.

Все файлы проекта сохраняются в той папке, которая сформировалась после указания в *поле* *Location* имени проекта (hello). На [рис. 1.19](https://intuit.ru/studies/courses/627/483/lecture/10944?page=4#image.1.19) показаны папки и файлы проекта *Visual Studio* 2010..



**Рис. 1.19.**Файлы и папки созданного проекта

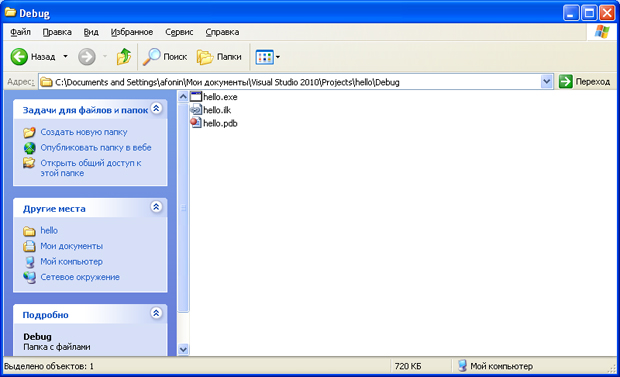
На [рис. 1.19](https://intuit.ru/studies/courses/627/483/lecture/10944?page=4#image.1.19) файлы с полученными расширениями означают:

**hello.sln** – *файл* решения для созданной программы. Он содержит информацию о том, какие проекты входят в данное решение. Обычно, эти проекты расположены в отдельных подкаталогах. Например, наш проект находится в подкаталоге hello;

**hello.suo** – *файл* настроек среды *Visual Studio* при работе с решением, включает информацию об открытых окнах, их расположении и прочих пользовательских параметрах.

**hello.sdf** – *файл* содержащий вспомогательную информацию о проекте, который используется инструментами анализа кода *Visual Studio*, такими как IntelliSense для отображения подсказок об именах и т.д.

Файлы папки **Debug** показаны на [рис. 1.20](https://intuit.ru/studies/courses/627/483/lecture/10944?page=4#image.1.20).



**Рис. 1.20.**Файлы папки Debug

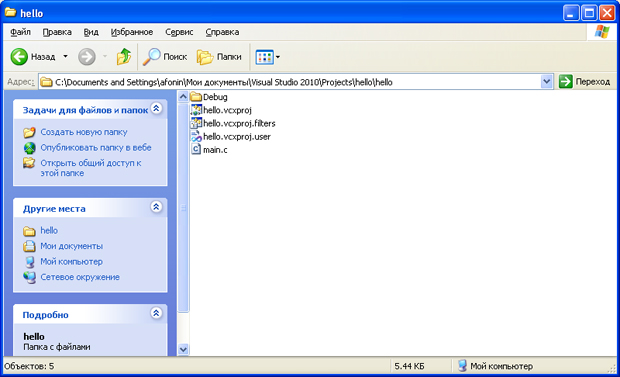
Рассмотрим файлы в соответствии с [рис. 1.20](https://intuit.ru/studies/courses/627/483/lecture/10944?page=4#image.1.20).

**hello.exe** – исполняемый *файл* проекта;

**hello.ilk** – *файл* "*incremental* linker", используемый компоновщиком для ускорения процесса компоновки;

**hello.pdb** – отладочная *информация*/*информация* об именах в исполняемых файлах, используемая отладчиком.

Файлы папки **hello** показаны на [рис. 1.21](https://intuit.ru/studies/courses/627/483/lecture/10944?page=4#image.1.21).



**Рис. 1.21.**Содержимое папки hello

Характеристика содержимого папки **hello**:

**main.c** – *файл* исходного программного кода,

**hello.vcxproj** – *файл* проекта,

**hello.vcxproj.user** – *файл* пользовательских настроек, связанных с проектом,

**hello.vcxproj.filters** – *файл* с описанием фильтров, используемых *Visual Studio* *Solution Explorer* для организации и отображения файлов с исходным кодом.

### Практическая часть

В практической части выполните следующие задания на основе рассмотренной программы **hello**:

1. Напишите программу, которая выводила бы на консоль название факультета, где учитесь, номер группы, свою фамилию, имя и отчество в разных строках дисплея (консоли) с помощью одной функции printf().
2. Вывод выполните с помощью нескольких функций printf() (количество функций должно соответствовать каждой порции информации).
3. Для задания пункта 2 вывод информации выполните в различных строках подряд, т.е. без межстрочного пропуска.
4. Проверьте программу без ключевого слова void для функции main().

**Примечание**. *Вывод* требуемой информации осуществляется с помощью букв латинского алфавита. Комментарии в программе могут быть сделаны после символа "//" или внутри комбинации символов "/\* \*/".

### **Контрольные вопросы**

1. Какие компиляторы языка **С** вам известны?
2. Какое имя имеет исполняемый файл созданного проекта?
3. Объясните назначение заголовочных файлов stdio.h, conio.h.
4. Как будет работать программа без заголовочного файла conio.h?
5. В каком месте программы находится точка ее входа?
6. Как осуществляется табуляция строки на консоли и на сколько позиций выполняется отступ от левого края?
7. Какое значение имеет главная функция проекта main() в программах на языке **С**?

***Лекция №2***

<https://professorweb.ru/my/programs/visual-studio/level1/1_2.php>

**Работа с Visual Studio**

Система Microsoft Visual Studio является самой сложной интегрированной средой разработки (integrated development enviroment - IDE), доступной для программистов в настоящее время. Она является результатом долгой истории развития языков программирования и интерфейсов и вобрала в себя достижения многих сред разработки программного обеспечения.

В этом руководстве мы подробно опишем систему Visual Studio, расскажем ее историю и покажем, что она предлагает разработчикам. Для примеров мы будем использовать версию Visual Studio 2013 (абсолютное большинство примеров можно будет проработать и в более ранних версиях этой программы и в новой версии Visual Studio 2015).

Независимо от выбранных установок в центре экрана вы увидите окно Start Page. Однако содержание этого окна и окружающих его панелей инструментов и окон может изменяться.

Перед тем как переходить к созданию своего первого приложения, стоит вернуться назад и просмотреть компоненты Visual Studio 2013 IDE. Меню и панели инструментов располагаются в верхней части экрана, а окна второго уровня, или панели (panes), появляются в левой, правой и нижней части главного окна. В центре находится главное окно редактирования: каждый раз, когда вы открываете исходный файл, XML-документ, форму или какой-нибудь другой файл, их содержание появляется в этом окне редактирования.

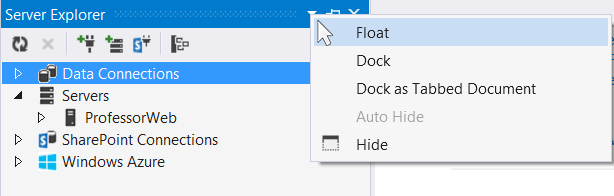
Одновременно с открытием файлов создается новая закладка, позволяющая переключаться между открытыми файлами. Окно редактирования окружено, набором инструментальных окон, обеспечивающих пользователя дополнительной информацией и функциональностью. Если пользователь выбрал стандартные установки, то на экране по умолчанию отображаются окна Solution Explorer и Class View в правой части и окна Server Explorer и Toolbox в левой части.

Инструментальные окна в левой части экрана находятся в свернутом (collapsed), или открепленном (unpinned), состоянии. Если щелкнуть на заголовке инструментального окна, оно откроется; когда фокус перемещается на другие компоненты или пользователь переводит курсор в другую область экрана, инструментальные окна снова сворачиваются.

Если хотите, чтобы инструментальное окно оставалось в развернутом (expanded), или закрепленном (pinned), состоянии, щелкните на средней пиктограмме, которая выглядит как булавка. Эта булавка повернется на 90 градусов, чтобы показать, что окно теперь закреплено. Щелкнув на пиктограмме со знаком "x", вы закроете окно. Если впоследствии захотите повторно открыть это или другое инструментальное окно, выберите его в меню View.

К некоторым инструментальным окнам, таким как окна потоков и контрольных выражений, нет доступа через меню View. В большинстве случаев доступ к этим окнам открывается с помощью альтернативных меню; например, доступ к окнам, связанным с отладкой программ, открывается с помощью меню Debug.

На рисунке ниже, показано контекстное меню, появляющееся после щелчка на первой пиктограмме (стрелке, направленной вниз):

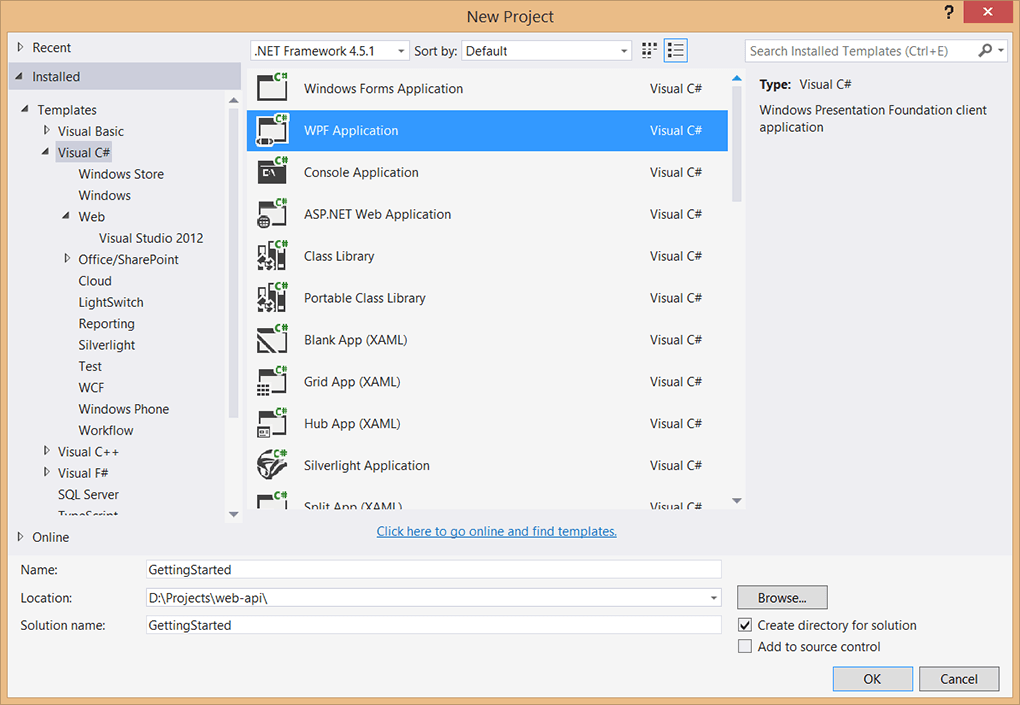


Каждый пункт в этом списке отражает отдельный способ настройки инструментального окна. Как легко догадаться, *команда Float* позволяет разместить инструментальное окно в любом месте экрана, независимо от главного окна интегрированной среды разработки. Это удобно, если у вас есть несколько экранов, поскольку в этом случае можно переместить разные инструментальные окна на дополнительный экран и открыть окна редактирования настолько широко, насколько это возможно. Выполнив *команду Dock as Tabbed Document*, можно создать дополнительную закладку для инструментального окна в окне редактирования.

Разработка, создание, отладка и развертывание вашего первого приложения

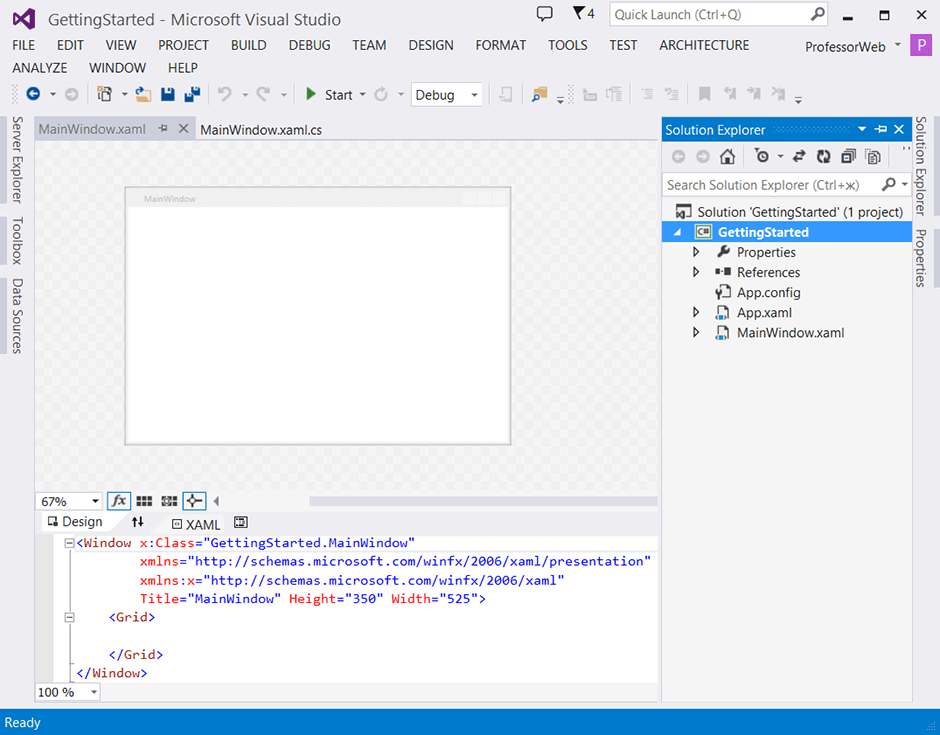
Завершив краткий обзор интегрированной среды разработки Visual Studio 2013, рассмотрим процедуру создания простого приложения, которая продемонстрирует работу с некоторыми из упомянутых компонентов. Разумеется, этим приложением будет сакраментальная программа "Hello, World", о которой должен знать каждый разработчик. Ее можно написать как на языке Visual Basic .NET, так и на языке C#. Это зависит от того, на каком из этих языков вам удобнее работать.

1. Начнем с команды File --> New --> Project, которая открывает диалоговое окно New Project, как показано на рисунке ниже:



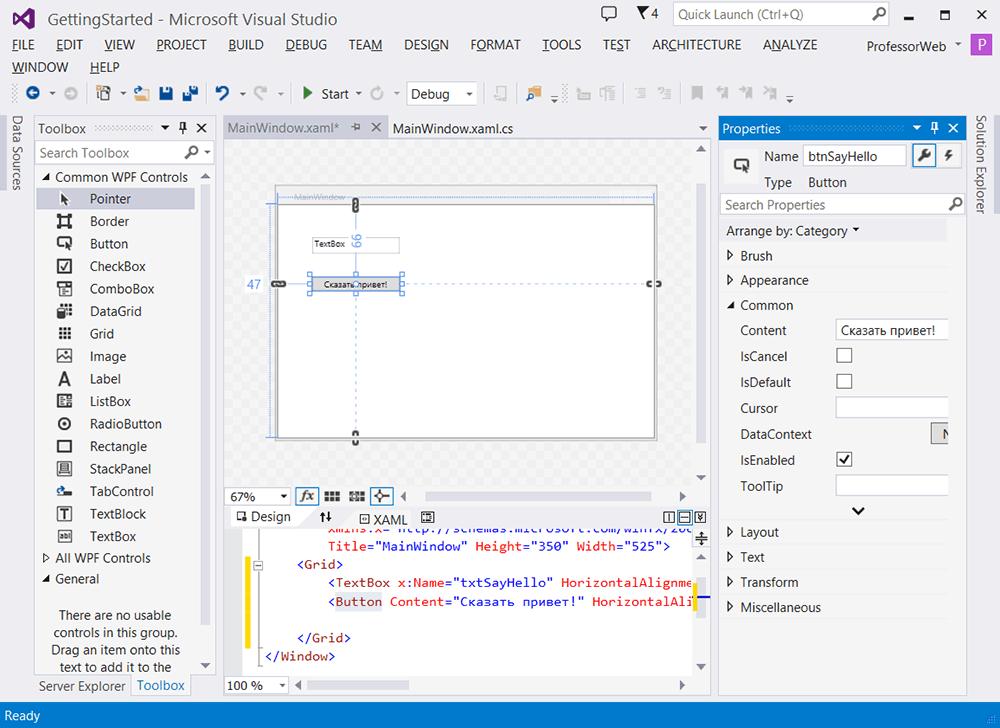
Если вы уже работали с более ранними версиями системы Visual Studio, то обнаружите, что это диалоговое окно подверглось значительному изменению. Оно по-прежнему содержит дерево в левой части диалогового окна, группирующего шаблоны по языкам программирования и технологиям, но теперь в правом верхнем углу есть поле поиска. Правая панель в этом диалоговом окне содержит дополнительную информацию о выбранном шаблоне проекта. И наконец, пользователь может выбрать версию платформы .NET. Каркас для этого приложения выбирается с помощью выпадающего меню в верхней части диалогового окна.

Выберите команду WPF Application в окне Templates (она расположена под корневыми узлами Visual Basic и Visual C# или под дочерним узлом Windows) и в списке Name выберите пункт GettingStarted, прежде чем щелкнуть на кнопке OK. В результате должен отобразиться новый проект приложения WPF, включающий в себя отдельное окно запуска и содержащийся в решении GettingStarted, как показано в окне Solution Explorer на рисунке ниже:



Это окно запуска автоматически открывается в окне визуального конструктора, чтобы пользователь имел представление о внешнем виде окна приложения во время его запуска.

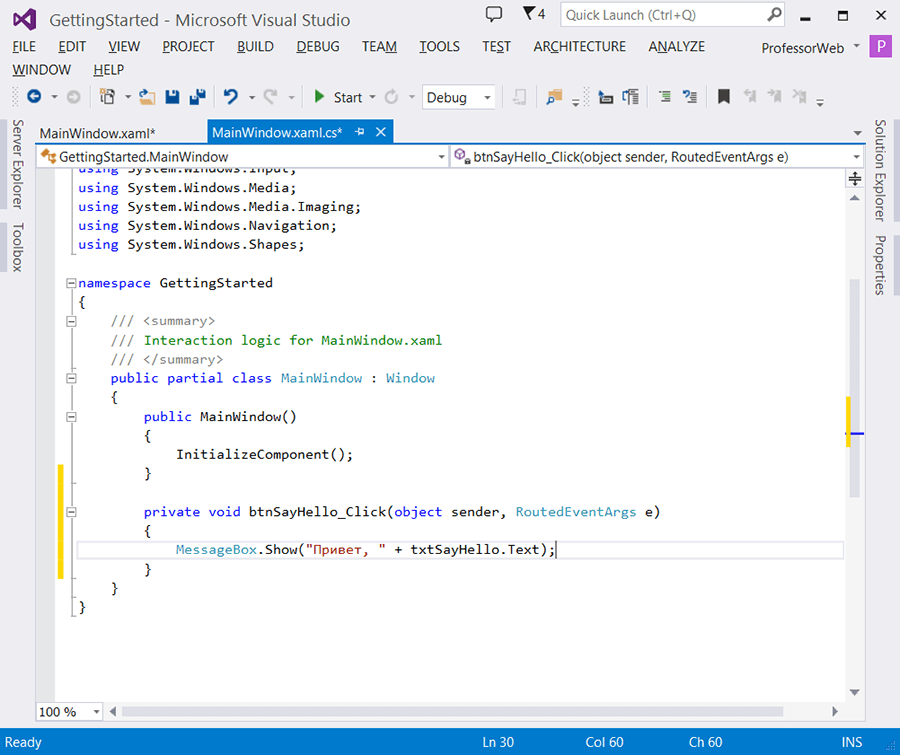
1. Щелкните на инструментальном окне Toolbox. В результате откроется окно с пиктограммой в виде булавки, фиксирующей его в открытом состоянии. Для того чтобы поместить в окно элементы управления, выберите соответствующие элементы в списке Toolbox и перетащите их на форму. В качестве альтернативы можете дважды щелкнуть на элементе управления, и система Visual Studio автоматически добавит его в окно.
2. Добавьте на форму кнопки и текстовые поля, чтобы она выглядела так, как на рисунке ниже. Выберите текстовое поле и инструментальное окно Properties (для того чтобы автоматически открыть окно Properties, можно нажать клавишу F4). Измените название текстового поля на txtToSay. Повторите эту процедуру для элемента Button, назвав ее btnSayHello и установив для свойства Content значение "Сказать привет".



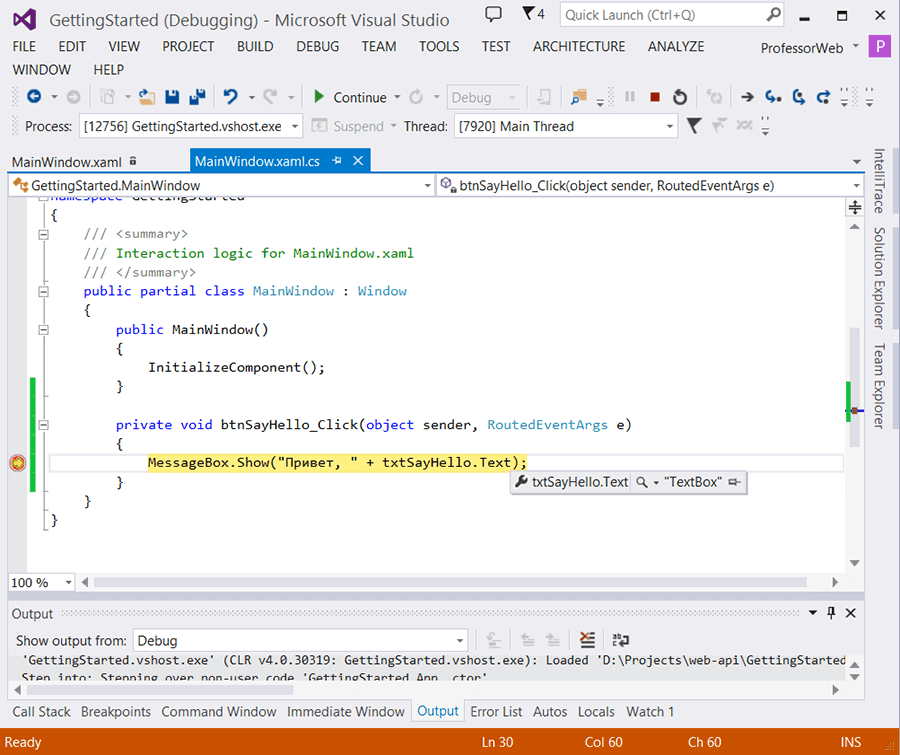
Любое свойство можно быстро найти, набрав его имя в поле поиска, расположенном под полем Name. Обратите также внимание на то, что после добавления элементов управления в окно после текста соответствующего пункта появляется звездочка (\*), означающая, что в этом пункте внесены изменения, которые еще не сохранены. Если вы попытаетесь закрыть этот пункт, не сохранив изменений, то система попросит вас сделать это. При создании приложения любой не сохраненный файл автоматически сохраняется в ходе процесса конструирования решения.

Следует помнить, что некоторые файлы, например файл решения, модифицируются, когда вы вносите изменения в среде Visual Studio 2013, без уведомления пользователя. Если вы попытаетесь выйти из приложения или закрыть решение, система предложит вам сохранить эти изменения.

1. Отмените выбор всех элементов управления, а затем дважды щелкните на кнопке. В результате не только откроется окно редактирования с кодом формы, но и будет создан и связан с приложением обработчик щелчков на кнопке. На рисунке ниже показано окно с кодом формы после вставки одной строки, выводящей на экран сообщение.

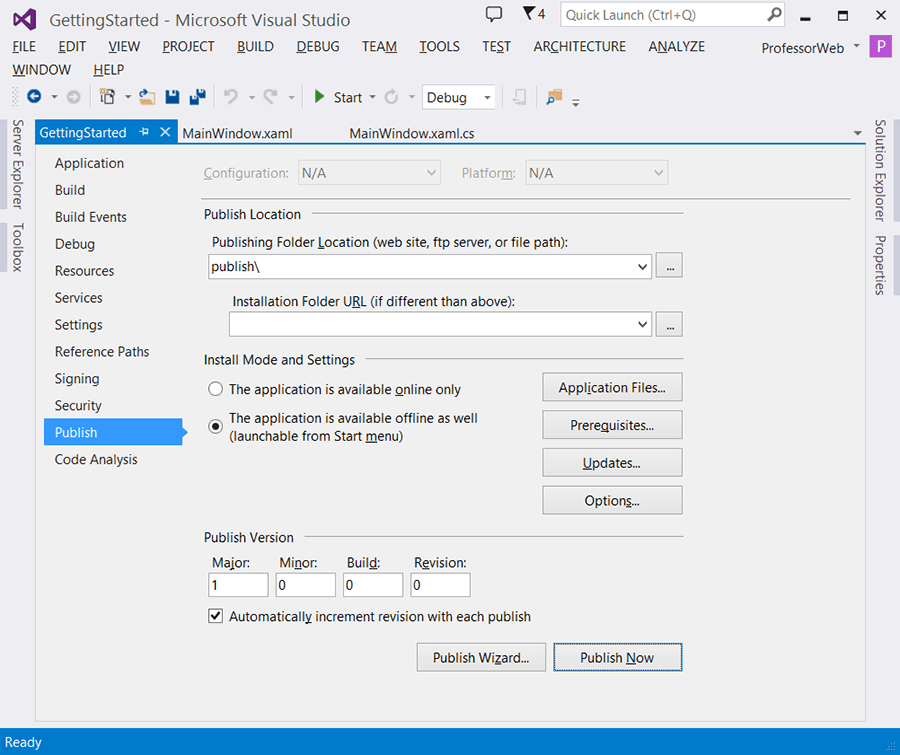


1. Перед тем создать и выполнить приложение, установите курсор в строке, содержащей имя MessageBox.Show, и нажмите клавишу F9. Это позволит вам установить *контрольную точку (breakpoint)*. Когда вы запустите приложение, нажав клавишу F5, а затем щелкнете на кнопке "Сказать привет" выполнение программы будет прервано в этом месте. Момент достижения контрольной точки показан на рисунке ниже. *Подсказка (data tip)*, появляющаяся на экране в момент, когда курсор мыши находится над указанной строкой, представляет собой содержание свойства txtToSay.Text.



Окно Visual Studio на рисунке значительно отличается от предыдущих окон, поскольку в его нижней части видны многочисленные новые инструментальные окна, а в верней части показаны новые командные панели. Если вы остановите приложение, то система Visual Studio восстановит предыдущие окна. Система Visual Studio 2013 поддерживает две разные схемы размещения объектов; во время проектирования и во время выполнения приложения. Меню, панели и окна но умолчанию имеют разный вид, когда вы редактируете проект и когда выполняете приложение. Можете изменить эти настройки по своему вкусу, и система Visual Studio 2013 запомнит их.

1. На последнем этапе происходит развертывание приложения. Независимо от того, какое приложение вы создаете - Windows Forms, WPF или веб, - система Visual Studio 2013 позволяет опубликовать ваше решение. Дважды щелкните на узле Properties в окне Solution Explorer и выберите узел Publish, чтобы показать варианты публикации вашего приложения, как показано на рисунке:



На рисунке показан путь к локальному каталогу для публикации, но вы можете также выбрать сетевую папку, каталог или сайт FTP. Выбрав место, в котором вы можете опубликовать свое решение, щелкните на кнопке Publish Now, и ваше решение будет опубликовано в указанном месте.

Итак, вы ознакомились с тем, как взаимодействуют разнообразные компоненты системы в процессе создания приложения. Перечислим типичные этапы создания решения:

1. Используйте меню File для создания решения.
2. Воспользуйтесь окном Solution Explorer, чтобы найти файл, который необходимо отредактировать, и дважды щелкните на нем, чтобы он появился в окне рабочего пространства.
3. Перетащите необходимые компоненты из окна Toolbox в окно приложения.
4. Поочередно выберите окно и каждый компонент, а также отредактируйте их свойства в окне Properties.
5. Дважды щелкните на окне или на элементе управления, чтобы получить доступ к коду, лежащему в основе их графического интерфейса.
6. Используйте основное рабочее пространство, чтобы написать код и разработать графический интерфейс, переключаясь между ними с помощью закладок, расположенных вверху окна.
7. Используйте инструментальные панели для запуска программы.
8. Если возникла ошибка, просмотрите окна Error List и Output.
9. Сохраните проект с помощью инструментальной панели или меню команд и выйдите из системы Visual Studio 2013.

В последующих статьях вы узнаете, как точнее настроить интегрированную среду разработки для своих потребностей и как система Visual Studio 2013 выдвигает множество предположений в ходе процесса разработки приложения. Вы также ознакомитесь с многими наиболее эффективными приемами работы с системой Visual Studio 2013, которые можно применить на практике.