Частное учреждение образования

«Колледж бизнеса и права»

|  |
| --- |
| УТВЕРЖДАЮ  Заведующий  методическим кабинетом  \_\_\_\_\_\_\_\_\_\_\_\_\_Е.В.Фалей  « \_\_\_\_ » \_\_\_\_\_\_\_\_\_ 2016 |

|  |  |
| --- | --- |
| Специальность: 2-40 01 01 «Программное обеспечение информационных технологий» | Учебная практика по программированию |
| Составлена на основании с учебной программой, утвержденной директором Колледжа бизнеса и права 31.05.2016 | |

**Практическая работа №3**

**Инструкционно-технологическая карта**

Тема:Обзор интегрированной среды разработки Microsoft Visual Studio. Типы проектов. Кодирование выражений и утверждений.

Цель: Закрепить общие сведения об интегрированной среде разработки Microsoft Visual Studio. Научиться создавать простейшие проекты и проводить исследование «скелета» кода. Закрепить умение кодировать выражения и утверждения.

Время выполнения: 2 часа

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Ознакомиться с теоретическими сведениями.
2. Установить Visual Studio на свой ноутбук или ПК (дома), начиная с версии 2012 и выше.
3. Ознакомиться с типами окон. Создать свой макет окон.
4. Создать проект типа ConsoleApplication. Закодировать выражения и утверждения согласно выданного преподавателем варианта индивидуального задания.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

**Обзор интегрированной среды разработки Microsoft Visual Studio**

Microsoft Visual Studio 2015 — 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. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.

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. Можно 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, например Unity и Apache Cordova, с помощью расширений [Набор средств Visual Studio для Unity](https://msdn.microsoft.com/ru-ru/library/dn940019.aspx) и [инструментов Visual Studio для Apache Cordova](https://msdn.microsoft.com/ru-ru/library/dn771545.aspx) соответственно. Можно самостоятельно расширить Visual Studio, создав собственные инструменты для выполнения специализированных задач.

Если раньше не использовали Visual Studio, с основами можно ознакомиться в учебниках и пошаговых руководствах [6], [8].

Описание новых возможностей Visual Studio 2015 см. [9].

**Установка Visual Studio**

Найти подходящий выпуск Visual Studio можно в разделе Выпуски Visual Studio [10].

Можно установить Visual Studio 2015, загрузив эту среду со страницы [Загружаемые файлы Visual Studio](https://www.visualstudio.com/downloads/download-visual-studio-vs.aspx) [11].

**Основы среды IDE**

На рисунке 1 показана интегрированная среда разработки (IDE) Visual Studio с открытым проектом и окном обозревателя решений для навигации по файлам проекта, а также окном Team Explorer для перемещения в системе управления версиями и отслеживания рабочих элементов. Указанные функции в заголовке окна более подробно описываются ниже.

**Вход**

При первом запуске Visual Studio можно выполнить вход с использованием учетной записи Майкрософт или рабочей учетной записи. Вход позволяет обеспечить синхронизацию параметров, например макетов окон, на нескольких устройствах и автоматическое подключение к нужным службам, таким как подписки Azure и Visual Studio Team Services. Если используете лицензию на основе подписки, то потребуется регулярно выполнять вход в Visual Studio для обновления токена лицензии. При наличии лицензии с ключом продукта выполнять вход необязательно, но рекомендуется для упрощения подключения к Visual Studio Team Services и учетным записям Salesforce.com, Azure и Office 365.

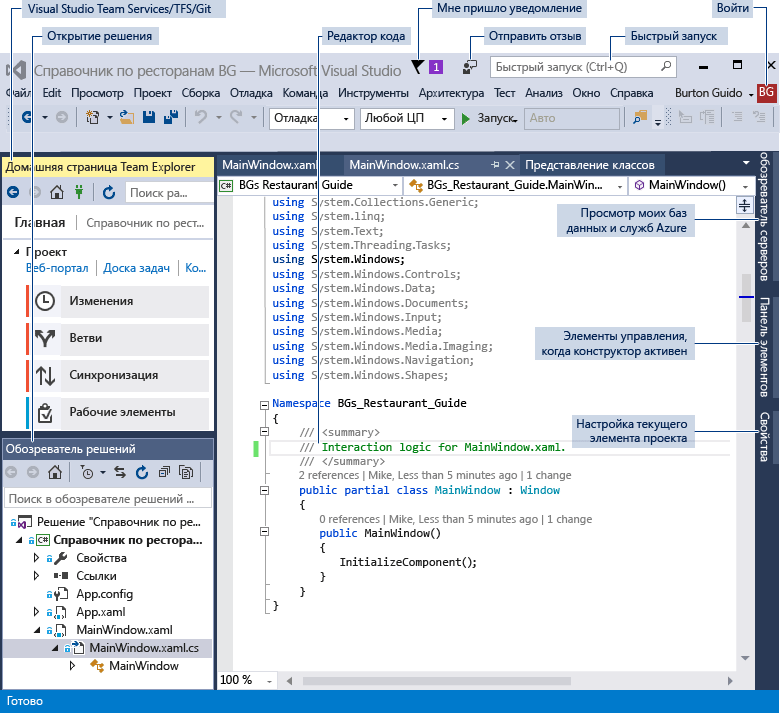


Рисунок 1

**Обновления**

Значок уведомления в правом верхнем углу заголовка окна указывает на наличие обновлений для Visual Studio или связанных компонентов, установленных в системе. Можно закрыть уведомление или выполнить требуемое действие.

**Поиск и получение справки**

Панель [быстрого запуска](https://msdn.microsoft.com/ru-ru/library/hh417697.aspx), показанная ниже на рисунке 2, дает возможность быстрого поиска команд, инструментов, функций и других компонентов Visual Studio, если вам неизвестно сочетание клавиш или расположение меню. Просто введите название нужного компонента, и на панели быстрого запуска появится ссылка на него.

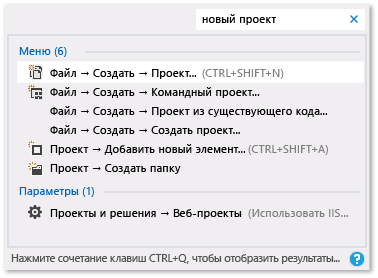


Рисунок 2

MSDN — это веб-сайт технической документации Майкрософт; страница, которую вы читаете прямо сейчас, находится на сайте MSDN. В Visual Studio можно нажать клавишу **F1**, чтобы перейти на страницу справки MSDN для активного окна. Кроме того, можно нажать клавишу **F1** в редакторе кода, чтобы перейти на страницу справки MSDN для API или ключевого слова, в котором в данный момент находится курсор. Например, в файле C# можно поместить курсор в любом месте или в конце объявления System.String и нажать клавишу **F1**, чтобы перейти на страницу справки MSDN по [String](https://msdn.microsoft.com/ru-ru/library/system.string.aspx).

**Персонализация среды IDE (управление окнами)**

Можно настроить макет окна среды разработки так, как вам удобно. Любое окно в любое время можно закрепить, открепить или скрыть, кроме того, редактор можно запускать в полноэкранном режиме. Можно создавать и сохранять различные настраиваемые макеты окон и работать только с теми окнами, которые требуются в определенном контексте. Например, можно создать полноэкранный макет, чтобы на экран выводился только редактор кода. Кроме того, можно создать разные макеты для отладки и для операций команды.

**Модификация настройки среды после установки**

В Visual Studio можно настроить расположение, размер и поведение окон, что позволит создать макеты окон, оптимально подходящие для различных рабочих процессов разработки. После того как макет будет настроен, интегрированная среда разработки запомнит его. Например, если после изменения места прикрепления окна **обозревателя решений** закрыть Visual Studio, во время следующего запуска окно **обозревателя решений** будет закреплено в том же месте, даже если работа будет продолжена на другом компьютере. Настраиваемому макету также можно присвоить имя, сохранить его, а затем переключаться между макетами с помощью одной команды. Например, можно создать один макет для редактирования, а другой —для отладки и переключаться между ними с помощью команды меню **Окно | Применить макет окна**.

## Типы окон

### Окна инструментов и документов

В среде IDE существует два основных типа окон: окна инструментов и окна документов. В число окон инструментов входят обозреватель решений, обозреватель серверов, окно вывода, список ошибок, конструкторы, окна отладчика и т. д. Окна документов содержат файлы исходного кода, произвольные текстовые файлы, файлы конфигурации и т. д. Изменять размер и перетаскивать окна инструментов можно с помощью их строки заголовка. Перетаскивание окон документов осуществляется с помощью вкладки. Правой кнопкой мыши щелкните вкладку или строку заголовка, чтобы задать другие параметры для окна.

В меню **Окно** находятся параметры закрепления, перемещения и скрытия окон в интегрированной среде разработки. Щелкните правой кнопкой мыши вкладку или строку заголовка окна, чтобы просмотреть дополнительные параметры для этого конкретного окна. Одновременно можно отображать несколько экземпляров окон инструментов. Например, можно открыть несколько окон веб-браузера. Кроме того, можно создать дополнительные экземпляры некоторых окон инструментов, выбрав команду **Создать окно** в меню **Окно**.

### Вкладка предварительного просмотра (окна документов)

На вкладке "Предварительный просмотр" можно просматривать файлы в редакторе, не открывая их. Просмотреть файлы можно путем их выбора в **обозревателе решений**, во время отладки при заходе в файлы, с помощью параметра "Перейти к определению", а также при просмотре результатов поиска. Файлы для предварительного просмотра отображаются на вкладке в правой части вкладки документа. Файл становится доступным для редактирования при его изменении или выборе команды **Открыть**.

### Группы вкладок

Группы вкладок расширяют возможности управления ограниченной рабочей областью при работе с двумя или более открытыми документами в интегрированной среде разработки. Несколько окон документов и окон инструментов можно упорядочить в вертикальные или горизонтальные группы вкладок и затем без труда переносить документы из одной группы в другую.

### Разделение окон

Если в документе необходимо одновременно просмотреть или изменить два различных места, можно разделить окна. Чтобы разделить документ на две части с независимой прокруткой, выберите **Разделить** в меню **Окно**. Чтобы восстановить единое представление, выберите **Снять разделение** в меню **Окно**.

### Панели инструментов

Панели инструментов можно упорядочить путем перетаскивания или с помощью диалогового окна **Настройка**.

## Размещение и закрепление окон

Окна документов и окна инструментов можно закрепить так, чтобы задать для них расположение и размер в рамках интегрированной среды разработки, или же их можно сделать плавающими независимо от интегрированной среды разработки. Окна инструментов можно закрепить в любом месте интегрированной среды разработки. Некоторые окна инструментов можно закрепить в виде окон с вкладками в окне редактора. Окна документов можно зафиксировать в рамках окна редактора. Их также можно закрепить в текущем расположении в последовательности перехода. Можно закрепить несколько окон для совместного перемещения по интегрированной среде разработки или за ее пределами. Окна инструментов можно скрывать или сворачивать.

Существуют следующие способы размещения окон:

* закрепление окон документов в левой части вкладки;
* прикрепление окон в виде вкладок к области редактирования;
* закрепление окон инструментов по краю области интегрированной среды разработки;
* перемещение окно документов или окно инструментов по интегрированной среде разработки или за ее пределами;
* скрытие окон инструментов вдоль края интегрированной среды разработки;
* отображение окон на разных мониторах;
* сброс расположения окон до макета по умолчанию или до сохраненного настраиваемого макета.

Окна инструментов и документов могут быть упорядочены путем перетаскивания, с помощью команд меню **Окно** и щелчка строки заголовка окна правой кнопкой мыши.

*Примечание:* Отображаемые на компьютере имена или расположения некоторых элементов пользовательского интерфейса Visual Studio могут отличаться от указанных в следующих инструкциях. Это зависит от имеющегося выпуска Visual Studio и используемых параметров

### Закрепление окон

При щелчке и перетаскивании строки заголовка окна инструмента или вкладки окна документа появляется маркер в виде ромба. Во время операции перетаскивания, когда указатель мыши наведен на одну из стрелок в ромбе, будет отображена затененная область, показывающая место закрепления окна после отпускания кнопки мыши.

Чтобы переместить закрепляемое окно без прикрепления к месту, во время перетаскивания окна нажмите клавишу CTRL.

Чтобы вернуть окно инструмента или окно документа в последнее закрепленное расположение, нажмите клавишу **CTRL** во время выполнения двойного щелчка строки заголовка или вкладки окна.

На следующем рисунке показан маркер в виде ромба для окон документов, которые могут быть закреплены только в рамках области редактирования.

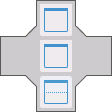


Рисунок 3

Окна инструментов можно прикрепить к одной стороне рамки в интегрированной среде разработки или в пределах области редактирования. Маркер в виде ромба (см рисунок 4) появляется при перетаскивании окна инструмента в другое место и упрощает процесс закрепления.

Маркер в виде ромба для окон инструментов

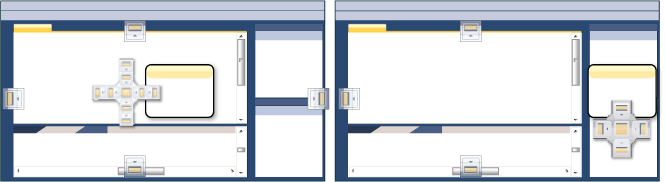


Рисунок 4

На следующем рисунке показан обозреватель решений, закрепленный в новом расположении, выделенном синим цветом.

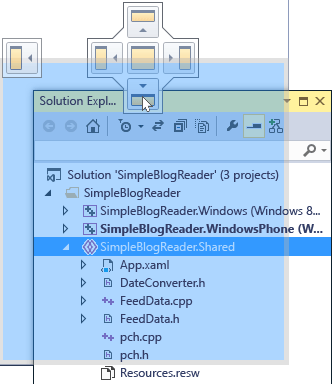


Рисунок 5

### Закрытие и автоматическое скрытие окон инструментов

Чтобы закрыть окно инструментов, нажмите кнопку "X" в верхнем правом углу строки заголовка. Для повторного открытия окна используется соответствующее сочетание клавиш или команда меню. Окна инструментов поддерживают возможность автоматического скрытия, позволяющую выводить окно из поля зрения при использовании другого окна. Если окно автоматически скрыто, его имя отображается на вкладке на краю интегрированной среды разработки. Для повторного использования окна наведите указатель на вкладку, в результате чего окно возвращается обратно в представление.

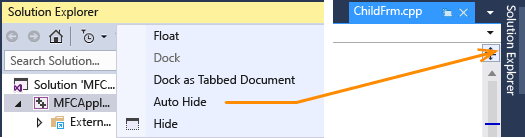


Рисунок 6

### Задание монитора

Если имеется второй монитор, который поддерживается операционной системой, можно выбрать, в каком мониторе будет отображаться окно. Можно даже сгруппировать несколько окон в блоки на других мониторах.

### Сброс, задание имен и переключение между макетами окон

Для возвращения исходного макета окон интегрированной среды разработки используется команда **Сбросить макет окон**. При запуске этой команды выполняются следующие действия.

* все окна перемещаются в расположения по умолчанию;
* окна, которые закрыты в макете по умолчанию, закрываются;
* окна, которые открыты в макете по умолчанию, открываются.

### Создание и сохранение пользовательских макетов

Visual Studio 2015 позволяет сохранять до 10 пользовательских макетов окон и быстро переключаться между ними. В приведенных ниже инструкциях показано, как создавать, сохранять, вызывать пользовательские макеты, использующие несколько мониторов как с закрепленными окнами средств, так и с перемещаемыми, а также управлять ими.

Сначала создайте тестовое решение с двумя проектами, каждый с отдельным оптимальным макетом.

##### **Создание проекта пользовательского интерфейса и настройка макета**

1. В диалоговом окне **Новый проект** создайте классическое приложение Visual C# WPF и назовите его. Представьте, что это и есть проект, в котором предстоит работать над пользовательским интерфейсом, вследствие чего для окна конструктора следует выделить максимальное пространство и закрыть остальные окна средства.
2. Если у вас несколько мониторов, переместите окна **Обозреватель сервера** и **Свойства** на второй монитор. В системе с одним монитором закройте все окна, кроме окна конструктора.
3. Нажмите клавиши **CTRL + ALT + X**, чтобы открыть панель элементов. Если окно закреплено, перетащите его в нужное место на любом мониторе.
4. Нажмите клавишу F5, чтобы включить в Visual Studio режим отладки. Настройте расположение окон отладки "Автоматические", "Стек вызовов" и "Выходные данные". Макет, который вы собираетесь создать, применит к ним режим редактирования и режим отладки.
5. Когда макеты будут открыты в режиме отладки и редактирования, в главном меню выберите пункт **Окно > Сохранить макет окна**. Назовите этот макет “Конструктор”.

Обратите внимание, что новому макету назначается сочетание клавиш из зарезервированного списка CTRL + ALT + 1...0.

##### **Создание проекта и макета базы данных**

1. Добавьте в решение новый проект **База данных SQL Server**.
2. Щелкните правой кнопкой мыши новый проект в обозревателе сервера и выберите **Просмотреть в обозревателе объектов**. Отобразится окно **Обозреватель объектов SQL Server**, в котором можно открывать таблицы, представления и другие объекты в базе данных. Это окно можно переместить или оставить закрепленным. Настройте другие окна средства. Для реалистичности можно добавить существующую базу данных, но это необязательно для данного пошагового руководства.
3. После настройки макета выберите в главном меню **Окно > Сохранить макет окна**. Назовите этот макет "Проект базы данных". (Для этого проекта не будет использоваться макет в режиме отладки.)

##### **Переключение между макетами**

Для переключения между макетами используйте сочетания клавиш или выберите в главном меню **Окно > Применить макет окна**.

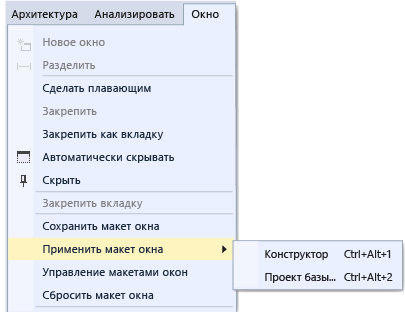


Рисунок 7

После применения макета пользовательского интерфейса обратите внимание на то, как выглядит макет в режиме редактирования и в режиме отладки.

Если установлено несколько мониторов на работе и один ноутбук дома, можно создать макеты, оптимизированные для всех компьютеров.

*Примечание:* Если макет для нескольких мониторов применить в системе с одним монитором, перемещающиеся окна на втором мониторе будут находиться за окном Visual Studio. Эти окна можно переместить на передний план, нажав клавиши ALT + TAB. Если позже открыть Visual Studio в системе с несколькими мониторами, можно восстановить окна в указанных расположениях, повторно применив макет.

##### **Управление макетами и перемещение их**

Пользовательский макет можно удалить, переименовать или переупорядочить, выбрав **Окно > Управление макетами окна**. Если переместить макет, привязка ключей будет настроена автоматически и отобразит новое положение в списке. Привязки невозможно изменить другим способом, таким образом, одновременно можно хранить не больше 10 макетов.

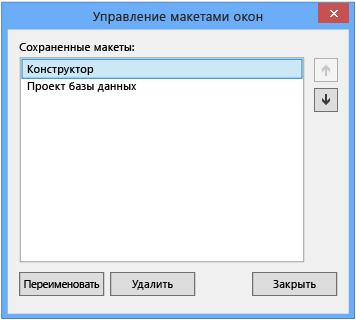


Рисунок 8

Чтобы вспомнить, какие сочетания клавиш назначены макетам, выберите **Окно > Применить макет окна**.

Сочетания клавиш предусмотрены практически для всех функций, и их также можно настроить. Чтобы создать новое сочетание клавиш, введите на панели быстрого запуска "Клавиатура", чтобы открыть диалоговое окно клавиатуры. После этого можно нажать клавишу F1, чтобы перейти на страницу справки MSDN, если требуются дополнительные сведения о параметрах.

**Создание проектов и решений**

Несмотря на то, что Visual Studio можно использовать для работы с отдельными файлами кода, как правило, работа выполняется в рамках *проекта*. Проект Visual Studio — это совокупность файлов и ресурсов, которые (для приложений) компилируются в единый двоичный исполняемый файл (например, EXE, DLL, APPX). Для веб-сайтов не на основе ASP.NET исполняемые файлы не создаются; проект содержит только HTML-код, файлы JavaScript и изображения. Поскольку иногда может потребоваться создать несколько двоичных файлов или веб-сайтов, которые тесно связаны, Visual Studio использует концепцию решения, которое может содержать несколько проектов или веб-сайтов. При создании проекта фактически создается проект в решении, что позволяет позднее добавить несколько проектов в это решение при необходимости. Например, если имеется проект библиотеки DLL, можно добавить в решение проект EXE, который будет загружать и использовать библиотеку DLL.

*Шаблон проекта* — это набор предварительно заполненных кодом файлов и параметров конфигурации, которые можно быстро настроить для создания приложения определенного типа. В комплект Visual Studio входит большое число шаблонов проектов, кроме того, если для ваших целей не подходит ни один из шаблонов по умолчанию, вы можете создать собственный. После создания проекта с помощью шаблона можно приступить к написанию собственного кода в имеющихся или в новых добавляемых файлах.

При создании в Visual Studio приложения, веб-сайта, веб-приложения, скрипта, подключаемого модуля и т. д. вы начинаете с проекта. С логической точки зрения проект содержит все файлы исходного кода, значки, изображения, файлы данных и прочие элементы, которые будут скомпилированы в исполняемую программу или веб-сайт, а также все остальное, что необходимо для выполнения компиляции. Проект также содержит все параметры компилятора и другие файлы конфигурации, которые могут потребоваться разным службам или компонентам, с которыми программа будет взаимодействовать.

В некотором смысле литерала проекта — это файл XML (\*.vbproj, \*.csproj, \*VCXPROJ-файл), определяющий иерархии виртуальной папки, а также пути для всех элементов, его «содержит» и все параметры построения. В Visual Studio файл проекта используется обозревателем решений для отображения содержимого и параметров проекта. При компиляции проекта подсистема MSBuild использует файл проекта для создания исполняемого файла. Можно также настроить проекты для создания выходных данных другого типа.

С точки зрения логики и файловой системы проект содержится в рамках решения, которое может содержать один или несколько проектов вместе с информацией о сборке, параметрами окна Visual Studio и любыми прочими файлами, которые не связаны с каким-либо проектом. Фактически решение является текстовым файлом в собственном уникальном формате; его обычно не изменяют вручную.

Решение включает связанный SUO-файл, в который входят параметры, предпочтения и сведения о конфигурации для каждого пользователя, работавшего над проектом.

На следующей схеме показана связь между проектами и решениями, а также элементы, которые они логически содержат.

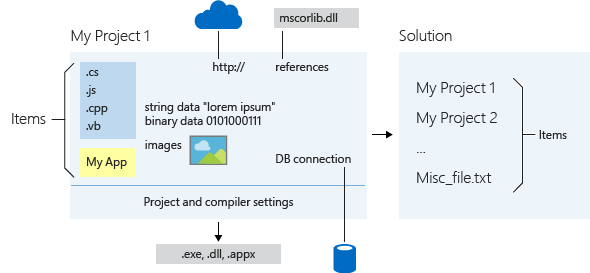


Рисунок 9

**Типы проектов**

Самый простой способ создать новый проект — начать с существующего шаблона проекта, который состоит из базового набора предварительно созданных файлов кода, файлов конфигурации, активов и параметров для создания приложения или веб-сайта определенного типа на конкретном языке программирования. Эти шаблоны, отображаемые в **диалоговое окно нового проекта** при выборе **файл | Создать | Проект** или **файл | Создать | Веб-сайт** в главном меню, а затем переходить.

## Управление проектами в обозревателе решений

После создания нового проекта вы используете **обозреватель решений** для просмотра проектов и решений, их связанных элементов, а также для управления ими. На рисунке 10 показан обозреватель серверов с решением C#, включающим два проекта.

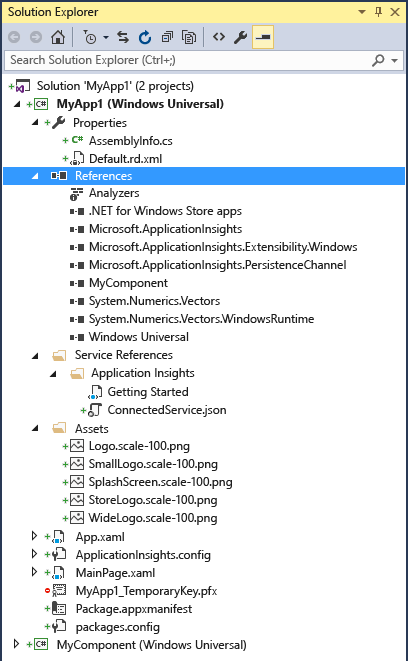


Рисунок 10

На рисунке 11 показано диалоговое окно нового проекта с шаблонами проектов, которые доступны для приложений ASP.NET.

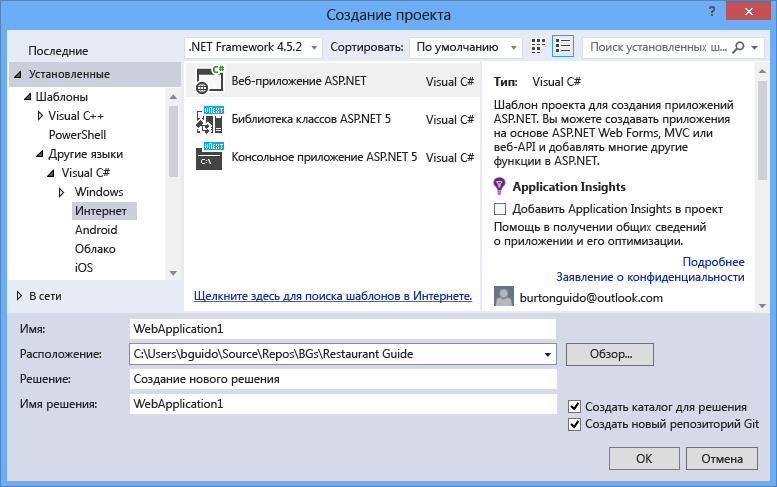


Рисунок 11

**Проектирование пользовательского интерфейса**

Конструктор — это удобный инструмент, который позволяет создавать пользовательский интерфейс без написания кода. Можно перетаскивать элементы управления пользовательского интерфейса, такие как поля со списками, календари и кнопки, из [панель элементов](https://msdn.microsoft.com/ru-ru/library/2381cd09.aspx) в рабочую область конструирования, представляющую окно или диалоговое окно. Можно изменять размер и расположение элементов без написания кода. Конструкторы включены для всех типов проектов, имеющих пользовательский интерфейс.

Если проект содержит пользовательский интерфейс на основе XAML, конструктор по умолчанию — Blend для Visual Studio, сложный графический инструмент, эффективно работающий в среде Visual Studio.

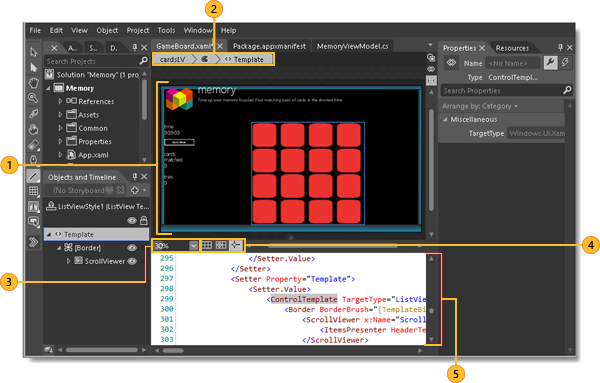


Рисунок 12

|  |  |
| --- | --- |
| Description | **Представление конструктора**. Содержит визуальную структуру документа. В этом представлении можно рисовать и изменять объекты на поверхности разработки. |
| Description | **Навигатор**. Позволяет быстро перемещаться между режимом редактирования шаблона, режимом редактирования стиля и областью редактирования объекта для выбранного объекта. |
| Description | **Масштаб**. Служит для изменения масштаба поверхности разработки и объектов на ней. |
| Description | **Элементы управления поверхности разработки**. Эти элементы управления (**Показать сетку привязки**, **Привязывать к линиям сетки** и **Включить/отключить привязку к линиям привязки**) служат для задания параметров привязки. Привязку удобно использовать для выравнивания объектов друг относительно друга или для их равномерного распределения на поверхности разработки. |
| Description | Правка **Редактор кода**. Служит для работы с кодом XAML, C#, C++ или Visual Basic вручную. |

**Написание, навигация и понимание кода**

Если вы являетесь разработчиком, вероятно, большую часть времени вы будете работать с окном редактора. Visual Studio включает редакторы для языков C#, C++, Visual Basic, JavaScript, XML, HTML, CSS и F#, а также сторонние подключаемые редакторы (и компиляторы) для многих других языков.

Отдельные файлы можно изменить в текстовом редакторе, последовательно выбрав **Файл | Открыть | Файл.** Чтобы изменить файлы в открытом проекте, щелкните имя файла в обозревателе решений. Код выделяется цветом. Для настройки цветовой схемы введите "Цвета" на панели быстрого запуска. Одновременно можно открыть большое число окон текстового редактора в виде вкладок. Кроме того, можно выполнить разбиение индивидуально для каждого окна. Текстовый редактор также можно запустить в полноэкранном режиме.

Текстовый редактор включает много интерактивных функций (если они требуются) и функций повышения производительности, помогающих ускорить написание кода. Функции различаются в зависимости от языка, и необязательно использовать их все (введите "Редактор" на панели быстрого запуска, чтобы включить или отключить функции). Некоторые распространенные возможности повышения производительности приведены ниже.

1. [Рефакторинг](https://msdn.microsoft.com/ru-ru/library/mt125495.aspx) включает такие операции, как интеллектуальное переименование переменных, перемещение выделенных строк кода в отдельную функцию, перемещение кода в другие расположения, изменение порядка параметров функции и т. д.
2. IntelliSense — это общий термин для набора очень популярных функций, отображающих сведения о типах в коде непосредственно в редакторе и в некоторых случаях, автоматически создающих небольшие отрывки кода. По сути IntelliSense представляет собой базовую документацию, встроенную в редактор, что избавляет от необходимости поиска информации о типах в отдельном окне справки. Функции IntelliSense зависят от языка.
3. Волнистые линии предупреждают об ошибках или потенциальных проблемах в коде в режиме реального времени по мере ввода, что позволяет исправлять их немедленно, не дожидаясь обнаружения ошибок во время компиляции или выполнения. Если навести указатель мыши на волнистую линию, на экран будут выведены дополнительные сведения об ошибке. Кроме того, в поле слева может появляться значок лампочки с предложениями по устранению ошибки.
4. [Закладки](https://msdn.microsoft.com/ru-ru/library/2wffxsy0.aspx) позволяют быстро переходить к определенным строкам в файлах, с которыми вы работаете.
5. В контекстном меню текстового редактора можно вызвать окно [Иерархия вызовов](https://msdn.microsoft.com/ru-ru/library/dd409859.aspx) для отображения методов, которые вызываются методом или вызывают метод, в котором установлен курсор.
6. **CodeLens** позволяет находить ссылки на код, изменения кода, связанные ошибки, рабочие элементы, проверки кода и модульные тесты — все это, не выходя из редактора.
7. Окно **Показать определения** позволяет просмотреть определение метода или типа в окне редактора, не покидая текущий контекст. Это окно теперь поддерживает и XAML-код.
8. Пункт контекстного меню **Перейти к определению** позволяет перейти непосредственно к тому месту, где определена функция или объект. Другие команды навигации также доступны по щелчку правой кнопкой мыши в редакторе.
9. Связанный инструмент, [обозреватель объектов](https://msdn.microsoft.com/ru-ru/library/exy1facf.aspx), позволяет исследовать сборки .NET или среды выполнения Windows в системе, просматривая содержащиеся в них типы (а также методы и свойства в этих типах).

Большинство пунктов меню "Правка" и "Вид" тем или иным образом связаны с редактором кода.

**Компилирование и сборка кода**

Сборка проекта означает компиляцию исходного кода и выполнение действий, необходимых для создания исполняемого файла. В разных языках предусмотрены разные операции сборки, а для обычных веб-сайтов сборка вообще не выполняется. Но независимо от типа проекта меню «Сборка» — это стандартное расположение этих команд. Чтобы скомпилировать и запустить код одним нажатием клавиши, воспользуйтесь клавишей F5. Все компиляторы можно настроить через среду IDE. Панель инструментов сборки позволяет указать, следует ли создавать отладочную версию программы с включенными символами и дополнительной проверкой на наличие ошибок для поддержки точек останова и пошагового режима отладчика или сборку выпуска, которую вы в итоге предоставляете клиентам. На странице свойств проекта можно настроить дополнительные параметры сборки и многие другие параметры. Щелкните правой кнопкой мыши узел проекта в обозревателе решений и выберите пункт "Свойства". Кроме того, сборку можно выполнить из командной строки.

Выходные данные сборки, включая сообщения об ошибках или успешном выполнении, отображаются в окне вывода. В окне "Список ошибок" приводятся подробные сведения об ошибках сборки.

[Окно интерпретации](https://msdn.microsoft.com/ru-ru/library/f177hahy.aspx) позволяет ввести выражение и сразу увидеть его результат.

[IntelliTrace](https://msdn.microsoft.com/ru-ru/library/dd264933.aspx) регистрирует все вызовы методов и другие события в работающей программе .NET и может помочь быстро найти источник проблемы.

**Консольные приложения**

При запуске консольного приложения операционная система создаёт так называемое консольное окно, через которое идёт весь ввод-вывод программ. Внешне это напоминает работу в операционной системе в режиме командной строки, когда ввод-вывод представляет собой поток символов.

Консольные приложения наилучшим образом подходят для изучения языка, так как в них не используется множество стандартных объектов, необходимых для создания графического интерфейса. Заготовка консольной программы выглядит следующим образом

using System;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Hello!!!");

}

}

}

Директива using System разрешает использовать имена стандартных классов из пространства имен System непосредственно без указания пространства. Ключевое слово namespace создает для проекта собственное пространство имен, названное по умолчанию ConsoleApplication1. Это сделано для того, чтобы можно было давать программным объектам имена, не заботясь о том, что они могут совпасть с именами в других пространствах имен.

В языке С# программа представляет собой совокупность взаимодействующих между собой классов. В программе автоматически создан класс с именем Program. Внутри класса – метод Main(), который обязан присутствовать в любом приложении – с него начинается выполнение программы.

Console – имя стандартного класса из пространства имен System. Его метод WriteLine() выводит на экран заданный в кавычках текст.

**Кодирование выражений и утверждений**

Вычисления, выполняемые кодом программы, осуществляются выражениями (expressions), которые представляют собой комбинации переменных, операторов (например, сложения или умножения) или ссылок на другие члены класса. В сле­дующем примере кода приведено выражение, которое осуществляет вычислитель­ную операцию и присваивает результат целой переменной. На C# такое выражение может выглядеть следующим образом: int result = 3 + 5 \* 7;

Переменная result в приведенном примере определена с использованием типа int. Назвать переменную можно как угодно; в данном примере выбрано имя result. Выражение 3 + 5 \* 7 представляет собой комбинацию операторов + (сложение) и \* (умножение). В ходе выполнения программы значение этого выражения вычис­ляется, а результат присваивается переменной result. Это значение будет равно 38, потому что выражения используют стандартные правила алгебраических при­оритетов. В рассматриваемом примере сначала вычисляется значение 5 \* 7, пото­му что приоритет имеет операция умножения. Результат умножения складывается с числом 3.

Чтобы изменить порядок операций при вычислении значения выражения, используются скобки. В следующем примере сначала выполняется сложение 3 и 5, а затем полученный результат умножается на 7:

int differentresult = (3 + 5) \* 7;

Вследствие группировки операндов при помощи скобок, переменная differentResult после вычисления выражения получит значение 56.

Тернарные операторы C# позволяют проверить истинность условия и, в зависимости от его соблюдения или несоблюде­ния, возвратить другое значение.

int bankAccount = 0;

string accountString = bankAccount == 0 ? "checking" : "savings";

Утверждения, выполняющие ветвления (branching), позволяют выбрать путь, по которому программа будет выполняться дальше, в зависимости от конкретного ус­ловия. Для этой цели применяются два основных типа утверждений ветвления: if и switch.

Утверждения if позволяют выполнить операцию только в том случае, когда логическое выражение, представляющее собой условие, в процессе выполнения программы получает значение true. Рассмотрим пример, в котором на консоль выводится утверждение, если переменная result получает значение больше, чем 48, в результате использования оператора сравнения > (greather than). В C# фигурные скобки не являются обязательными, если используется только одно утверждение, которое должно быть выполнено после того, как условие станет истин­ным (получит логическое значение true). Однако использование фигурных скобок станет обязательным, если при соблюдении условия выполнены должны быть два или большее количество выражений. Условие при этом должно получить одно из двух логических (Boolean) значений — либо true, либо false. Кроме того, здесь можно использовать раздел else, который выполняется в случае несоблюдения усло­вия. Раздел (clause) — это еще один метод указать, что конкретный элемент является частью другого, составного выражения. Ключевое слово else не используется в ка­честве выражения как такового, поэтому оно и называется разделом (clause).

if (result > 48)

{

Console.WriteLine("result is > 48");

}

else

{

Console.WriteLine("result is <= 48");

}

Утверждение switch сообщает программе о необходимости вычислить одно из множества условий и, в зависимости от результата, осуществить ветвление.

switch (name){

case "Joe": Console.WriteLine("Name is Joe"); break;

case "Megan": Console.WriteLine("Name is Megan"); break;

default: Console.WriteLine("Unknown Name"); break;

}

ВАРИАНТЫ ИНДИВИДУАЛЬНЫХ ЗАДАНИЙ

Определить, принадлежит ли точка М (х,у) множеству D, где D – заштрихованная на рисунке область.

Номер варианта определяется по формуле

НомерВарианта = (НомерСтроки – 1) х 4+НомерСтолбца

|  | 1 | 2 | 3 | 4 |
| --- | --- | --- | --- | --- |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |

ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Дейтел, П. Как программировать на Visual C# 2012 / П. Дейтел, Х. Дейтел. – 5-е изд. – СПб.: Питер, 2014. – 864 с.: ил.
2. Майо, Дж. Самоучитель Microsoft Visual Studio 2010 / Дж. Майо. – СПб.: БХВ-Петербург, 2011. – 464 с.: ил.
3. Стиллмен, Э. Изучаем C# / Э. Стиллмен, Дж. Грин. – 3-е изд. – СПб.: Питер, 2014. – 816 с.: ил.
4. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – 6-е изд. – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.: ил.
5. Шилдт, Г. C#4.0: Полное руководство / Г. Шилдт. – М.: ООО «И.Д. Вильямс», 2011. – 1056 с.: ил.
6. Шарп Джон. Microsoft Visual C#. Подробное руководство / Джон Шарп. – 8-е изд. – СПб.: Питер, 2017. – 848 с.
7. Полное руководство по языку программирования С# 6.0 и платформе .NET 4.6 [Электронный ресурс] / Сайт о программировании. – metanit.com, 2012-2016. – Режим доступа: [http://metanit.com/sharp/ tutorial](http://metanit.com/sharp/%20tutorial). – Дата доступа: 20.08.2016.

# Начало работы (Get Started Developing with Visual Studio) [Электронный ресурс] / Microsoft. – Microsoft, 2016. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/ms165079.aspx. – Дата доступа: 20.08.2016.

# Новые возможности Visual Studio 2015 [Электронный ресурс] / Microsoft. – Microsoft, 2016. – Режим доступа: https://msdn.microsoft.com/ru-ru/library/bb386063.aspx. – Дата доступа: 20.08.2016.

1. Интегрированная среда разработки [Электронный ресурс] / Microsoft. – Microsoft, 2016. – Режим доступа: <https://www.visualstudio.com/>ru/vs/. – Дата доступа: 20.08.2016.
2. Загружаемые файлы Visual Studio [Электронный ресурс] / Microsoft. – Microsoft, 2016. – Режим доступа: <https://www.visualstudio.com/>ru/downloads/. – Дата доступа: 20.08.2016.

Преподаватель Е.В. Багласова

|  |
| --- |
| Рассмотрено на заседании цикловой  комиссии программного обеспечения информационных технологий № 10  Протокол №\_\_\_\_\_от «\_\_\_»\_\_\_\_2016  Председатель ЦК *( )* Т.Г.Багласова |