Visual Studio 2013/15

DVO group©

МЕтодические указания по работе

ГАПОНОВ В.О. Карпунин Н.В.

2016

[1. Введение 3](#_Toc455660804)

[2. Меню 3](#_Toc455660805)

[2.1. Основные 3](#_Toc455660806)

[2.1.1. «Файл» 3](#_Toc455660807)

[2.1.2. «Правка» 4](#_Toc455660808)

[2.1.3. «Вид» 5](#_Toc455660809)

[2.1.4. «Проект» 5](#_Toc455660810)

[2.1.5. «Сборка» 5](#_Toc455660811)

[2.1.6. «Отладка» 5](#_Toc455660812)

[2.1.7. «Команда» 7](#_Toc455660813)

[2.1.8. «Сервис» 7](#_Toc455660814)

[2.1.9. «Тест» 7](#_Toc455660815)

[2.1.10. «Архитектура» 7](#_Toc455660816)

[2.1.11. «Анализ» 7](#_Toc455660817)

[2.1.12. «Окно» 7](#_Toc455660818)

[2.1.13. «Справка» 7](#_Toc455660819)

[2.2. Добавление, удаление или перемещение меню в строке меню 7](#_Toc455660820)

[3. Элементы управления 9](#_Toc455660821)

[3.1. Панели управления 9](#_Toc455660822)

[3.2. Панель инструментов 11](#_Toc455660823)

[3.3. Настройки панели инструментов 11](#_Toc455660824)

[3.4. Добавление нового меню 14](#_Toc455660825)

[4. Горячие клавиши 15](#_Toc455660826)

[4.1. Файл 15](#_Toc455660827)

[4.2. Правка 15](#_Toc455660828)

[4.3. Правка-Дополнительно 16](#_Toc455660829)

[4.4. Правка-Закладки 16](#_Toc455660830)

[4.5. Правка-Структура 16](#_Toc455660831)

[4.6. Вид 17](#_Toc455660832)

[4.7. Вид - Другие окна 17](#_Toc455660833)

[4.8. Проект 18](#_Toc455660834)

[4.9. Сборка 18](#_Toc455660835)

[4.10. Отладка 18](#_Toc455660836)

[4.11. Сервис 18](#_Toc455660837)

[4.12. Архитектура 18](#_Toc455660838)

[4.13. Анализ 19](#_Toc455660839)

[4.14. Справка 19](#_Toc455660840)

[5. Создание консольных приложений 20](#_Toc455660841)

[5.1. Создание проекта 20](#_Toc455660842)

[5.2. Рабочая область проекта и добавление файлов исходного кода. 21](#_Toc455660843)

[5.3. Написание консольного приложения на примере программы «Автомагазин» на языке C++ 23](#_Toc455660844)

[6. Отличия Visual Studio 2013 от версии 2015 года 31](#_Toc455660845)

[6.1. Выбор целевой платформы. 31](#_Toc455660846)

[6.2. Универсальные приложения для любого устройства на Windows 10. 31](#_Toc455660847)

[6.3. Интернет 32](#_Toc455660848)

[6.4. .NET Framework 32](#_Toc455660849)

[6.5. C++ 32](#_Toc455660850)

[6.6. Строка меню предварительного просмотра устройств 32](#_Toc455660851)

[6.7. Диагностика графики 33](#_Toc455660852)

[6.8. Динамический анализ кода 33](#_Toc455660853)

[7. Создание приложений с использованием WPF 35](#_Toc455660854)

[7.1. Описание WPF 35](#_Toc455660855)

[7.2. Написание приложения 38](#_Toc455660856)

[7.3. Пример работы приложения 41](#_Toc455660857)

[8. Список источников 43](#_Toc455660858)

# Введение

Visual Studio 2013 используется для создания приложений с использованием языков программирования Visual C++, Visual C#, Visual Basic, Visual F# и т.д.

В данном методическом пособии будут рассмотрены Visual C++ для создания консольных приложений и Visual C# для создания приложений в оконном интерфейсе.

# Меню

## Основные

### «Файл»

Содержит элементы управления для создания, открытия, закрытия проекта или решения, а также для управления файлами в конкретном проекте или решении.

**«Создать»** - даёт к созданию проекта с выбором функционала, веб-сайта, командного проекта или отдельного файла.

**«Открыть»** - позволяет открыть файлы проекта с выбором функционала или веб-сайта.

**«Закрыть»** - закрывает файл, над которым ведётся работа.

**«Закрыть решение»** - закрывает текущее решение.

**«Сохранить»** и **«Сохранить всё»** - позволяет сохранить соответственно текущий файл или все файлы решения.

**«Печать»** - позволяет распечатать файлы проекта в формат PDF для последующей отправки на печать.

**«Последние файлы»** и **«Последние проекты и решения»** - позволяет открыть недавно закрытые файлы или решения.

**«Выход»** - закрытие Visual Studio.

### «Правка»

Используется для действий по редактированию кода, таких как вставка, копирование, удаление, поиск и замена.

**«Отменить»** и **«Вернуть»** - позволяет соответственно отменить или вернуть последнее действие.

**«Отменить последнее глобальное действие»** и **«Вернуть последнее глобальное действие»** - позволяет соответственно отменить или вернуть последнее действие глобальное действие (переименование класса или пространства имён, изменение в нескольких файлах).

**«Вырезать»**, **«Копировать»**, **«Вставить»** - позволяет соответственно вырезать, копировать или вставить часть кода.

**«Удалить»** - позволяет удалить выделенный фрагмент кода.

**«Выделить всё»** - позволяет выделить весь текст документа.

**«Поиск замена»** - предоставляет доступ к функциям нахождения фрагментов кода по заданному ключу и их замене.

**«Перейти»** и **«Перейти в…»** - позволяет соответственно перейти к заданной строке или к заданному слову в коде.

**«Вставить файл как текст»** - позволяет вставить содержимое текстового файла в код.

**«Дополнительно»** - предоставляет доступ к функциям по форматированию документа.

**«Закладка»** - предоставляет доступ к функциям по работе с закладками.

**«Intellisense»** - предоставляет доступ к функциям по работе с интеллектуальным помощником редактирования кода «Intellisence».

**«Выполнить рефракторинг»** - предоставляет доступ к функциям по рефракторингу кода (переименованию полей, переменных, классов и т.д.).

### «Вид»

Используется для настройки интерфейса Visual Studio. Функционал, связанный с панелями управления, описан в разделе [***Панели управления***](#_Панели_управления).

**«Панель инструментов»** – позволяет произвести настройки панели инструментов.

**«Во весь экран»** – позволяет отобразить окно редактора кода в полноэкранном режиме.

### «Проект»

Предоставляет функционал для работы с файлами проекта, к примеру, добавление или удаление файлов.

### «Сборка»

Предоставляет функционал для сборки, перестройки и анализа кода решения

### «Отладка»

Используется для настройки режима отладки приложения.

**«Окна»** - предоставляет доступ к дополнительным окнам Visual Studio.

**«Графика»** - предоставляет доступ к отладке графических приложений.

**«Начать отладку»** и **«Запустить без отладки»** - предоставляет доступ к управлению запуском проекта в режиме отладки.

**«Исключения»** - позволяет настроить список выдаваемых исключений.

**«Производительность и диагностика»** - предоставляет доступ к функциям по измерению расходов используемых приложением ресурсов.

**«Шаг с заходом»** и **«Шаг с обходом»** - позволяет соответственно производить отладку с заходом в выполняемую функцию или обойти её.

**«Точка останова»** - позволяет установить точку остановки исполнения проекта при отладке.

**«Создать точку останова»** - позволяет установить точку остановки в функции при отладке.

**«Отладка IntelliTrace»** - предоставляет доступ к функциям управления интеллектуальным помощником отладки «IntelliTrace»

**«Параметры и настройки…»** - предоставляет доступ к настройке параметров отладки.

**«Свойства…»** - позволяет изменить свойства проекта.

### «Команда»

Используется для подключения к средствам командной разработки приложений (в данной методичке работа с ней не рассматривается).

### «Сервис»

Используется для подключения баз данных, удалённых серверов, расширений и для более глубокой настройки Visual Studio.

### «Тест»

Даёт доступ к средствам для тестирования приложений.

### «Архитектура»

Используется для постройки карт кода и дерева зависимости классов приложения.

**«Создать схему»** - позволяет создать схему классов приложения (для проектов на платформе .NET Framework).

**«Сформировать диаграмму зависимостей»** - позволяет сформировать диаграмму зависимостей классов, пространств имён и библиотек.

### «Анализ»

Используется для анализа ресурсов, потребляемых приложением.

### «Окно»

Используется для настройки интерфейса Visual Studio.

### «Справка»

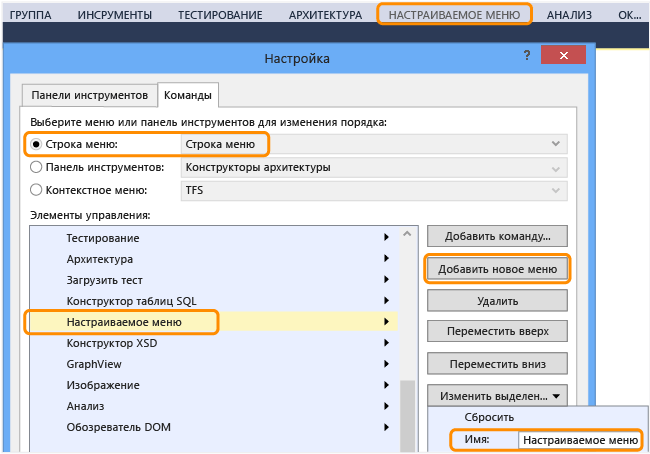
Предоставляет доступ к официальной документации по Visual Studio.

## Добавление, удаление или перемещение меню в строке меню

1. В строке меню выберите **Сервис**, **Настроить**.

Откроется диалоговое окно **Настройка**.

1. На вкладке **Команды** установите переключатель **Строка меню** и флажок **Строка меню** в списке рядом с этим параметром, а затем выполните одно из следующих действий:
   * Чтобы добавить меню, нажмите кнопку **Добавить новое меню**, нажмите кнопку **Изменить выбор** и введите имя меню, который требуется добавить.



* + Чтобы удалить меню, выберите его из списка **Элементы управления** и нажмите кнопку **Удалить**.
  + Чтобы переместить меню в строке меню, выберите данное меню в списке **Элементы управления**, а затем нажмите кнопку **Вверх** или **Вниз**.

# Элементы управления

## Панели управления

Для получения доступа к настройкам интерфейса Visual Studio с помощью панелей управления нужно выбрать пункт меню **«Вид».**

Их можно группировать путем перетаскивания друг на друга. Для ориентации на рабочей области используются всплывающие подсказки размещения.

Далее будет приведён список панелей и описание их функционала.

1. **Код (**по умолчанию**) –** отображает текст выбранного документа исходного кода.
2. **Вывод** **(**по умолчанию**) –** отображает сведения о процессе линковки и компиляции, также выводит сообщения о ходе выполнения программы.

«Очистить все» удаляет все строки вывода.

1. **Обозреватель решений (**по умолчанию**) –** позволяет просмотреть включаемы в решение проекты и файлы в этих проектах, внутреннее наполнение файлов (классы, функции, переменные т.д.).

Имеет поисковую строку для быстрого нахождения нужных файлов или папок, кнопки «свернуть все» и «показать все файлы», реализующие соответствующий функционал. Конопка «свойства» позволяет настроить параметры проекта, решения или отдельного файла.

1. **Список ошибок (**по умолчанию**) –** отображает предупреждения компилятора и список ошибок в программе.

При выводе ошибок, также выводится дополнительна информация в виде описание/файл/строка/столбец. IntelliSense выдаст подсказки для распространённых ошибок.

Для перехода к месту с ошибкой достаточно сделать двойной щелчок левой кнопкой мышки по нужной строке[[1]](#footnote-1).

1. **Командный обозреватель –** предоставляет доступ к тому же функционалу, что и обозреватель решения, а также предоставляет функционал для командной разработки.
2. **Обозреватель серверов** – позволяет посмотреть список подключённых серверов, позволяет подключить новые.
3. **Обозреватель архитектуры** – позволяет посмотреть список классов и файлов проекта.
4. **Обозреватель объектов SQL Server –** позволяет посмотреть подключённые к проекту базы данных.
5. **Окно закладок –** позволяет посмотреть закладки в файлах исходного кода.
6. **Иерархия вызовов –** позволяет посмотреть иерархию вызовов функций, переменных и т.д.
7. **Классы –** позволяет посмотреть классы проекта.
8. **Начальная страница –** открывает вкладку с начальной страницей.
9. **Список задач –** позволяет создавать и просматривать план работы.

**«Следующая задача»** и **«Предыдущая задача»** позволяют переключаться между текущими задачами.

1. **Панель элементов –** используется для создания приложений с использованием форм. Подробнее ее функционал будет рассмотрен в разделе [***Создание приложений с использованием WPF и Windows Forms***](#_Создание_приложений_с)**.**
2. **Уведомления –** позволяет посмотреть список уведомлений о наличии обновлений для Visual Studio.

## Панель инструментов

 «Переключение между недавними окнами». Если в процессе разработки вы работаете с несколькими окнами, то используйте эти клавиши для быстрой навигации между ними.

 «Сохранение или открытие файлов проекта». Используется для управления файлами проекта в удобном интерфейсе.

 «Переключение между последними изменениями в коде». Эти клавиши предназначены для отмены или, наоборот, возвращения сделанных изменений в исходном коде.

 «Запуск и отладка приложения». С помощью данных клавиш производится запуск проекта в конфигурациях отладки (Debug), выпуска (Release) или им подобных.

 «Пойск в файлах». Используется для поиска заданных файлов проекта.

 «Управление комментированием кода». Эти клавиши позволяют закоментировать или, наоборот, раскоментировать выделелнные блоки кода.

 «Закладки». С помощью этих клавиш можно создавать и редактировать закладки для быстрой навигации в файлах исходного кода.

## Настройки панели инструментов

При работе с инструментами разработки соответствующий им функционал панели управления будет подключен автоматически.

Для доступа к настройкам панели правления нужно зайти в **Вид-Панели управления** или нажать правой кнопкой мыши по панели, или левой кнопкой мыши (Рисунок 1) и из списка выбрать пункт **Параметры…** , в открывшимся окне (Рисунок 2) можно настроить панель и конкретные разделы.

Рисунок 1

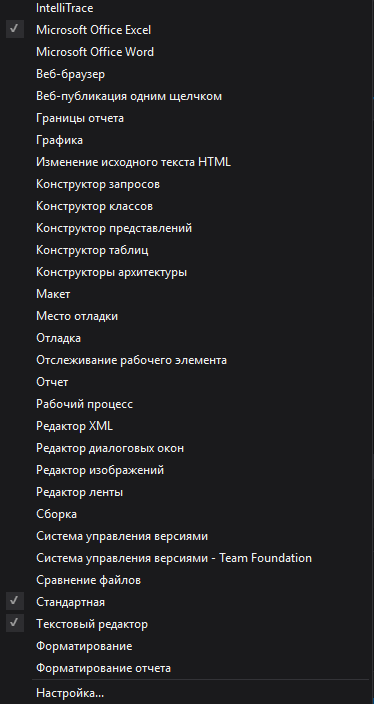
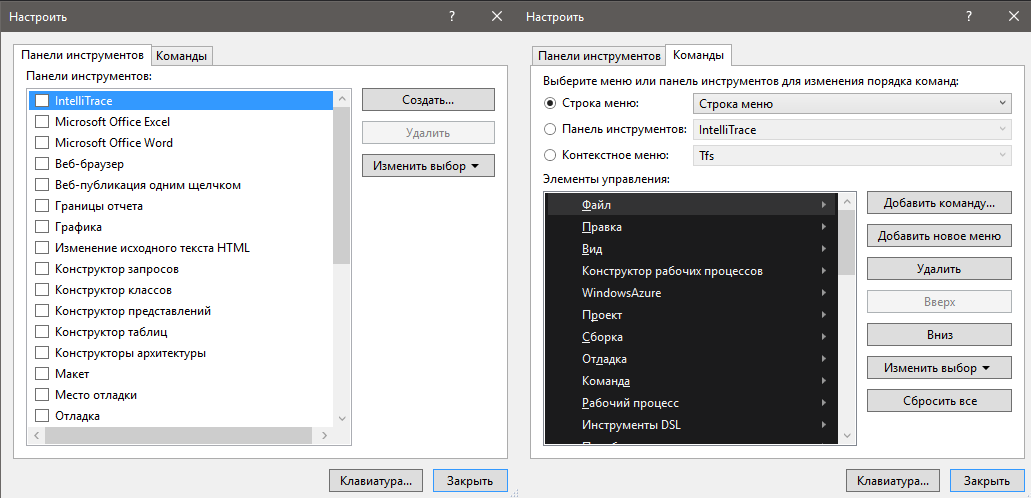


Рисунок 2



**Стандартные** и **Текстовый редактор –** стандартный набор кнопок, описанный в разделе [Панель инструментов](#_Панель_инструментов).

**Конструктор запросов –** добавляет функционал для составления запросов к базе данных.

**Конструктор представлений –** добавляет функционал для анализа базы данных.

**Конструктор таблиц –** добавляет функционал для составления таблиц базы данных.

**Конструктор архитектуры –** работает с UML конструктором, добавляет функционал для помощи в создании диаграмм.

**Конструктор классов –** добавляет функционал для редактирования классов при наличии карт кода.

**Отладка** – добавляет функционал для помощи в отладке программы.

**Отслеживание рабочего элемента –** добавляет функционал для распределения задач при командной разработке проектов.

**Редактор изображений –** добавляет функционал для удобного редактирования графических файлов.

**Сборка –** добавляет функционал для помощи в сборки программы.

**Система управления версиями –** добавляет функционал для удобной командной разработки.

**Сравнение файлов –** позволяет сравнивать изменения в файлах при командной разработке.

**Форматирование –** добавляет функционал для редактирования текстовых документов.

**IntelliTrace –** добавляет функционал для управления интеллектуальным помощником в отладке «IntelliTrace»

**Microsoft Office Excel –** добавляет функционал для удобного редактирования Excel-документов.

**Microsoft Office Word –** добавляет функционал для удобного редактирования Word-документов.

**Веб-браузер –** добавляет функционал для работы в сети «Интернет».

**Веб-публикации одним щелчком –** добавляет функционал для управления веб-сайтом.

**Графика –** добавляет функционал для разработки графических приложений.

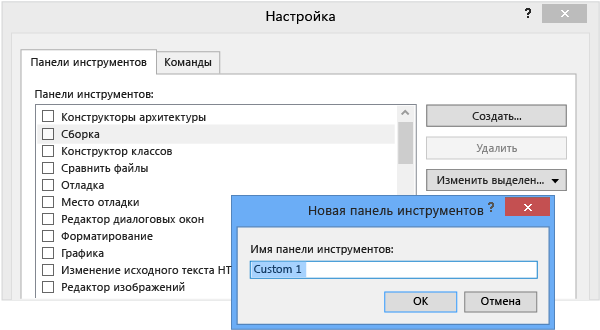
**Изменение исходного текста HTML –** добавляет функционал для редактирования HTML-документов.

## Добавление нового меню

1. В строке меню выберите **Сервис, Настроить**.

Откроется диалоговое окно **Настройка**.

1. На вкладке **Панель инструментов** выполните одно из нижеуказанных действий.
   * Чтобы добавить панель инструментов, нажмите кнопку **Создать**, укажите имя панели инструментов, которую требуется добавить, а затем нажмите кнопку **OK**.



* + Чтобы удалить пользовательскую панель инструментов, выберите его из списка **Панели инструментов** и нажмите кнопку **Удалить**.[[2]](#footnote-2)
  + Чтобы переместить панели инструментов в другое место закрепления, выберите его в списке Панели инструментов, нажмите кнопку «Изменить выбор», а затем выберите нужное местоположение в отобразившемся списке.

# Горячие клавиши

## Файл

**«Ctrl+N»** создать новый файл

**«Ctrl+Shift+N»** создать новый проект

**«Shift+Alt+ N»** создать новый веб-сайт

**«Ctrl+Shift+O»** открыть проект

**«Shift+Alt+ O»** открыть веб-сайт

**«Ctrl+S»** сохранить файл

**«Ctrl+Shift+S»** сохранить все файлы

**«Ctrl+P»** печать

**«Alt+F4»** выход

## Правка

**«Ctrl+C»** копировать выделенный участок

**«Ctrl+V»** вставить в указанное место

**«Ctrl+Shift+V»** вставить следующий фрагмент

**«Ctrl+Z»** вернуться на шаг назад

**«Ctrl+Y»** вернуться на шаг назад

**«Ctrl+X»** вырезать указанный участок

**«Ctrl+F»** найти указанный текст

**«Ctrl+H»** быстрая замена

**«Ctrl+Shift+F»** найти в файлах

**«Ctrl+Shift+H»** заменить в файлах

**«Ctrl+A»** выделить все

**«Ctrl+G»** перейти…

**«Ctrl+б»** перейти в…

**«Del»** удалить

## Правка-Дополнительно

**«Ctrl+K, Ctrl+D»** форматировать документ

**«Ctrl+K, Ctrl+F»** форматировать выделенный документ

**«Ctrl+Shift+U»** все прописные

**«Ctrl+U»** все строчные

**«Alt+стрелка вверх»** переместить выбранные строки вверх

**«Alt+стрелка вниз»** переместить выбранные вниз

**«Ctrl+K, Ctrl+\»** удалить пустое пространство по горизонтали

**«Ctrl+R, Ctrl+W»** показать пустое пространство

**«Ctrl+E, Ctrl+W»** переносить по словам

**«Ctrl+I»** последовательный поиск

**«Ctrl+K, Ctrl+C»** закомментировать выделенный фрагмент

**«Ctrl+K, Ctrl+U»** раскомментировать выделенный фрагмент

## Правка-Закладки

**«Ctrl+K, Ctrl+K»** закладка

**«Ctrl+K, Ctrl+P»** предыдущаязакладка

**«Ctrl+K, Ctrl+N»** следующаязакладка

**«Ctrl+K, Ctrl+L»** удалить все закладки

**«Ctrl+Shift+K, Ctrl+Shift+P»** предыдущая закладка в папке

**«Ctrl+Shift+K, Ctrl+Shift+N»** следующая закладка в папке

**«Ctrl+K, Ctrl+H»** добавить ярлык списка задач

## Правка-Структура

**«Ctrl+M, Ctrl+M»** свернуть/развернуть сегмент

**«Ctrl+M, Ctrl+L»** свернуть/развернуть все сегменты

**«Ctrl+M, Ctrl+P»** прекратить показ структуры

**«Ctrl+M, Ctrl+U»** прервать скрытие текущей области

**«Ctrl+M, Ctrl+O»** свернуть в определения

## Вид

**«F7»** код

**«Ctrl+Alt+L»** обозреватель кода

**«Ctrl+\, Ctrl+M»** командный обозреватель

**«Ctrl+Alt+S»** обозреватель серверов

**«Ctrl+\, Ctrl+R»** обозреватель архитектуры

**«Ctrl+\, Ctrl+S»** обозреватель объектов SQL Server

**«Ctrl+K, Ctrl+W»** окно закладок

**«Ctrl+Alt+K»** иерархия вызовов

**«Ctrl+Shift+C»** классы

**«Ctrl+\, D»** окно определения кода

**«Ctrl+Alt+J»** иерархия объектов

**«Ctrl+\, E»** список ошибок

**«Ctrl+Alt+O»** вывод

**«Ctrl+\, T»** список задач

**«Ctrl+Alt+X»** панель элементов

**«Ctrl+W, N»** уведомления

**«Shift+Alt+M»** все окна

**«Shift+Alt+ВВОД»** все окна

**«Ctrl+-»** назад

**«Ctrl+Shift+-»** вперёд

**«F4»** окно свойств

**«Shift+F4»** окна свойств

## Вид - Другие окна

**«Ctrl+Alt+A»** окно команд

**«Shift+Alt+D»** источник данных

**«Ctrl+Alt+R»** веб-браузер

**«Ctrl+\, Ctrl+U»** обозреватель моделей UML

**«Ctrl+Alt+T»** структура документа

**«Ctrl+Shift+E»** ресурсы

## Проект

**«Ctrl+Shift+A»** добавить новый элемент…

**«Shift+Alt+A»** добавить существующий элемент…

## Сборка

**«Ctrl+Shift+B»** собрать решение

**«Alt+F11»** анализ кода в решение[[3]](#footnote-3)

## Отладка

**«F5»** начать отладку

**«Ctrl+F5»** запуск без отладки

**«Ctrl+Alt+E»** исключения…

**«Alt+F2»** производительность и диагностика

**«F10»** шаг с обходом

**«F11»** шаг с заходом

**«F9»** точка остановки

**«Ctrl+Shift+F9»** удалить все точки остановки

## Сервис

**«Ctrl+K, Ctrl+B»** диспетчер фрагментов кода…

**«Ctrl+Alt+P»** присоединиться к процессу…

## Архитектура

**«Ctrl+\, Ctrl+N»** создать карту кода

## Анализ

**«Alt+F11»** выполнить анализ кода в решение

**«Alt+F2»** производительность и диагностика

## Справка

**«Ctrl+F1»** просмотрсправки

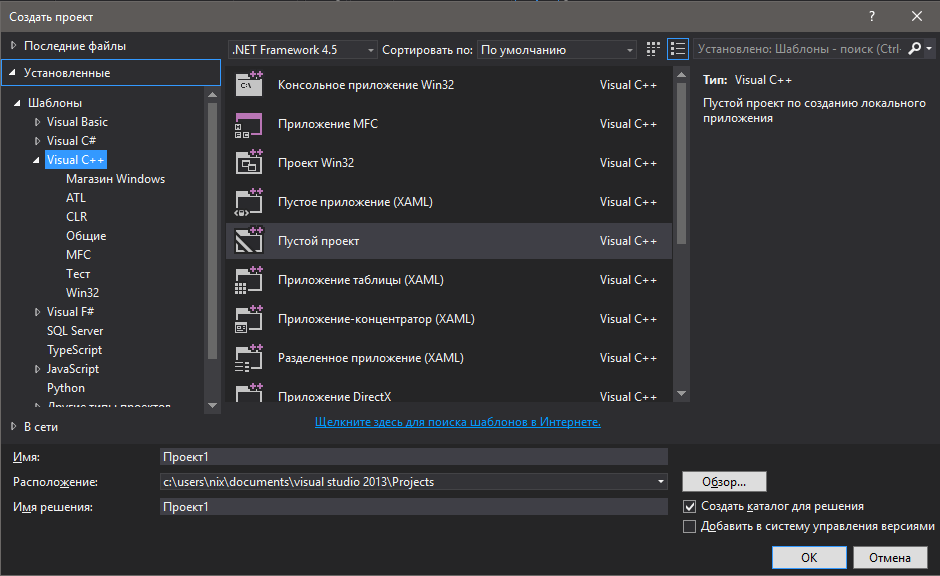
**«Ctrl+Alt+F1»** добавление и удаление содержимого справки

# Создание консольных приложений

## Создание проекта

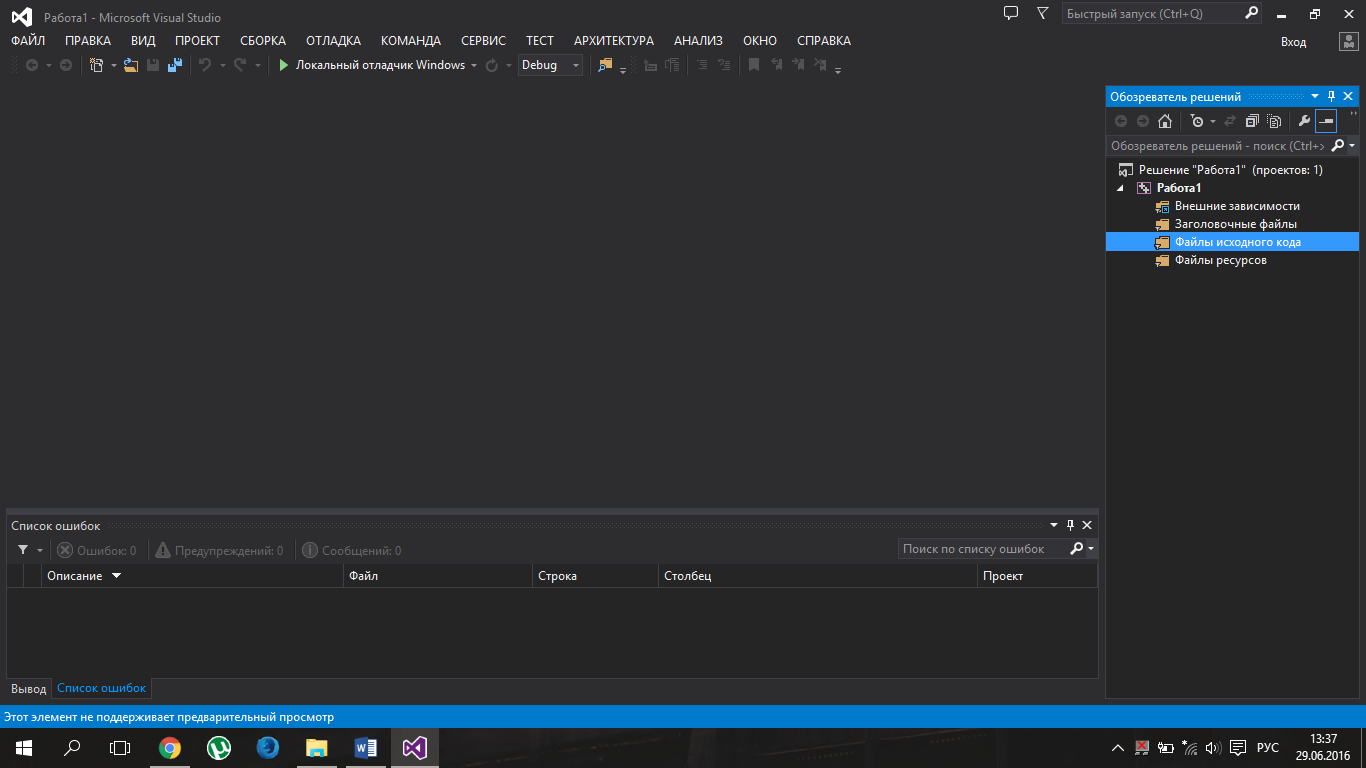
Открываем **Файл-Создать-Проект.** В появившемся окне (Рисунок 3) выбираем **Шаблон: Visual C++**, далее выбираем тип проекта, в нашем случае создадим **Пустой проект**, в поле **Имя**, **Расположение**, **Имя решения** выбираем соответственно имя проекта, его расположение и название решения в которое он входит. Также можно указать нужно ли создать для решения отдельную папку и включить ли его в систему управления версиями. После завершения всех настроек жмем кнопку **Ок.**

Рисунок 3

****

## Рабочая область проекта и добавление файлов исходного кода.

Рисунок 4

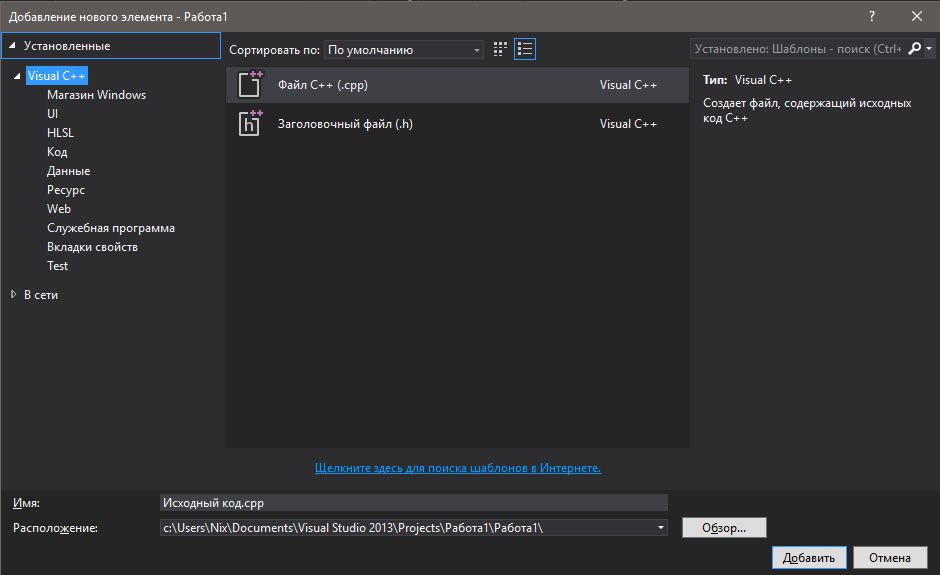


В рабочей области присутствуют все ранее описанные элементы, такие как **Обозреватель решений, Вывод, Список ошибок, Меню, Панель инструментов.**

Для создания файлов проекта необходимо нажать комбинацию клавиш «Ctrl+Shift+A» для нового файла или «Shift+Alt+A» для добавления существующего, либо через выпадающий список (правый клик по нужно папке) выбрать **Добавить-Новый/Существующий файл.**

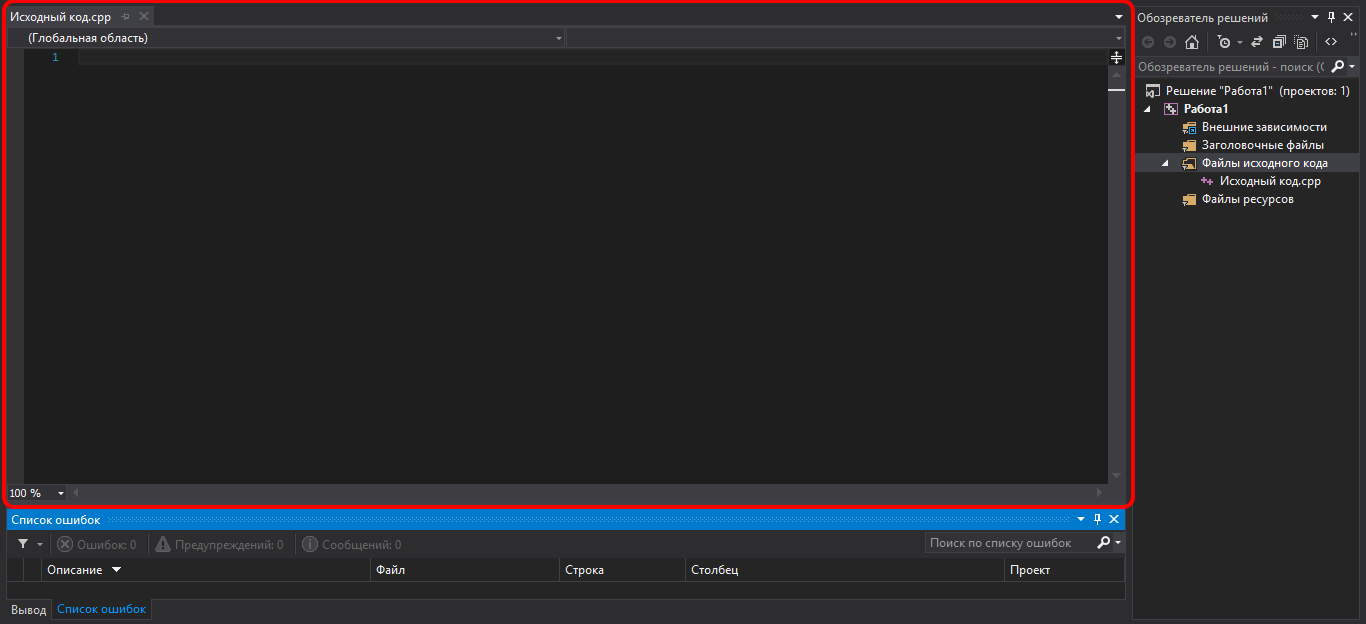
После предыдущих действий появится диалоговое окно (Рисунок 5) в которой можно выбрать имя добавляемого файла, его расположение и тип[[4]](#footnote-4). После всех настроек жмем **Добавить**.

Рисунок 5



Теперь мы можем начать написание кода программы в окне **Код** (красная область Рисунок 6)

Рисунок 6



## Написание консольного приложения на примере программы «Автомагазин» на языке C++

Далее будет рассмотрено создание приложения на языке C++ в качестве интерфейса приложения будет использована консоль ОС Windows. В процессе написания мы рассмотрим средства индикации ошибок синтаксиса, процесс их устранения и способы отладки приложения. Для выделения вставок исходного кода будет использован специальный шрифт (int main).

Определим класс «Car»

Car.h\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#ifndef CAR\_

#define CAR\_

#include<iostream>

#include<string>

class Car{

protected:

unsigned PowerEngine;

unsigned MaxSpeed;

float TimeHandred;

float FuelCons;

std::string TypeBody;

std::string FuelType;

std::string Mark;

std::string Model;

public:

void PrintInfo(){

std::cout << "\nMark: " << Mark

<< "\nModel: " << Model

<< "\nTypeBody: " << TypeBody

<< "\nPowerEngine: " << PowerEngine << " hp"

<< "\nMaxSpeed: " << MaxSpeed << " km/h"

<< "\nTimeHandred: " << TimeHandred << " sec"

<< "\nFuelCons: " << FuelCons << " liter per 100 km"

<< "\nFuelType: " << FuelType;

};

void EnteredInfo(){

std::cout << "\nEntered Mark: ";std::getline(std::cin, Mark);

std::cout << "\nEntered Model: "; std::getline(std::cin, Model);

std::cout << "\nTypeBody: "; std::getline(std::cin, TypeBody);

std::cout << "\nFuelType: "; std::getline(std::cin, FuelType);

std::cout << "\nFuelCons: "; std::cin >> FuelCons;

std::cout << "\nEntered Power of Engine: "; std::cin >> PowerEngine;

std::cout << "\nMaxSpeed: "; std::cin >> MaxSpeed;

std::cout << "\nTimeHandred: "; std::cin >> TimeHandred;

};

};

#endif

Создадим класс «Shop», который будет инкапсулировать в себе массив объектов класса «Car» и реализовывает функции добавления, удаления и вывода элементов.

Shop.h\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#ifndef SHOP\_

#define SHOP\_

#include"Car.h"

#include <vector>

#include <Windows.h>

class Shop {

private:

std::vector<Car> shop;

public:

void Add(){

Car newcarr;

newcarr.EnteredInfo();

shop.push\_back(newcarr);

std::cin.get();

}

void Remove(){

if (!shop.empty()){

unsigned N = 0;

std::cout << "Entered number of Note: ";

std::cin >> N;

if (N > 0 && N != 0 && N-1 < shop.size())

shop.erase(shop.begin() + N-1);

else{ std::cout << "Not correct number!"; Sleep(2000); return; }

}

}

void PrintCat(){

unsigned number = 1;

for each (Car car in shop)

{

std::cout<<"Number of Note: " << number++ <<std::endl;

car.PrintInfo();

std::cout << std::endl<<std::endl;

}

}

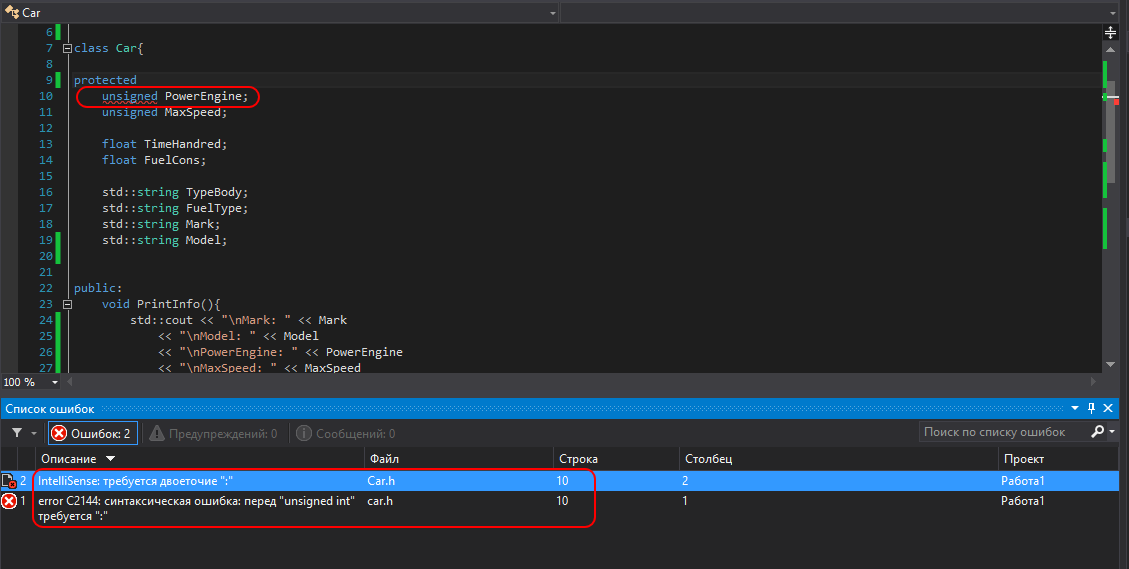
};

#endif

Предотвращает повторное включение в проект одних и тех же файлов с помощью директивы #include.

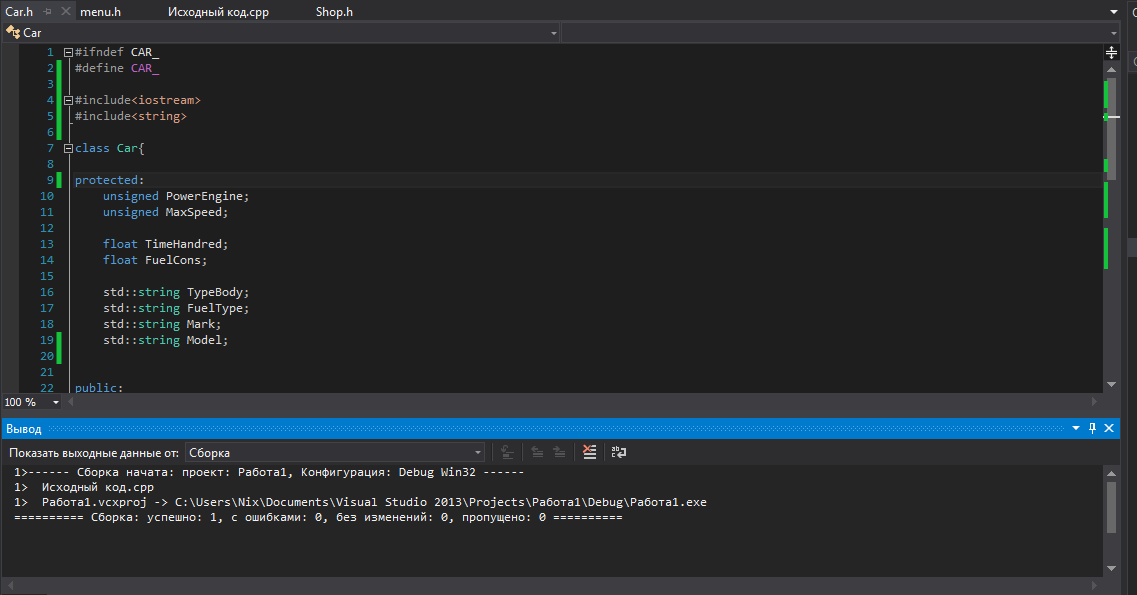
В процессе написания этих классов могут возникать ошибки. Пример такого случая праведен на Рисунок 7.

Рисунок 7



Мы не поставили «:» перед модификатором доступа класса. Исправим данную ошибку и посмотрим результат сборки проекта (Рисунок 8).

Рисунок 8



Как можно заметить ошибок больше нет и сборка завершена успешно.

Исходный код.cpp\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#include"Shop.h"

#include"исходники меню\menu.h"

int main()

{

Shop shop;

initializer\_list<Items> list\_init // инициализатор меню.

{

{ "Add", &shop, &Shop::Add },

{ "Remove", &shop, &Shop::Remove },

{ "Print", &shop, &Shop::PrintCat }

};

Menu someMenu("Main menu");

FillMenu(someMenu, FillVector(list\_init));

someMenu.MenuControl();

}

Ниже будет кратко описано устройство меню программы, за него отвечают три файла, но ввиду их большого размера они не будут тут приведены[[5]](#footnote-5).

1. Menu.h

Содержит структуру **Items** и определение класса **Menu**. С помощью них реализуется представление меню и подменю, подменю реализованы рекурсивно, как меню вложенное в предыдущие меню.

К меню можно подключить объект любого класса и связанные методы[[6]](#footnote-6), подробнее можно прочесть ниже

1. Menu.cpp

Содержит определение методов класса **Menu**.

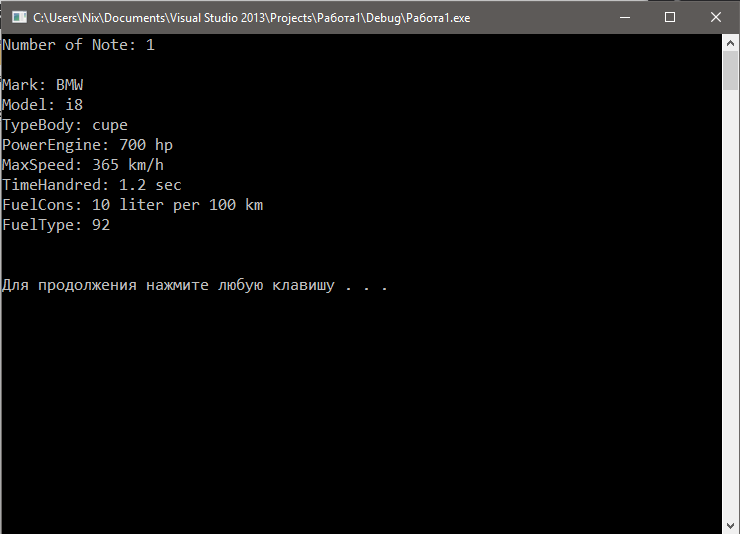
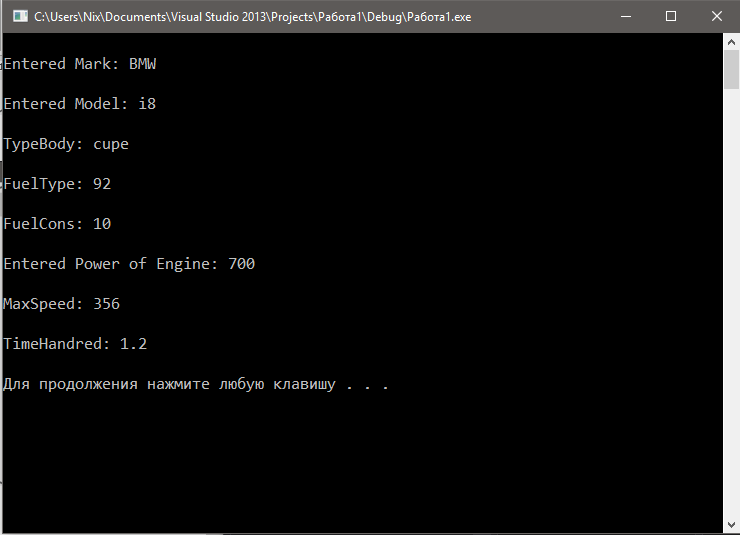
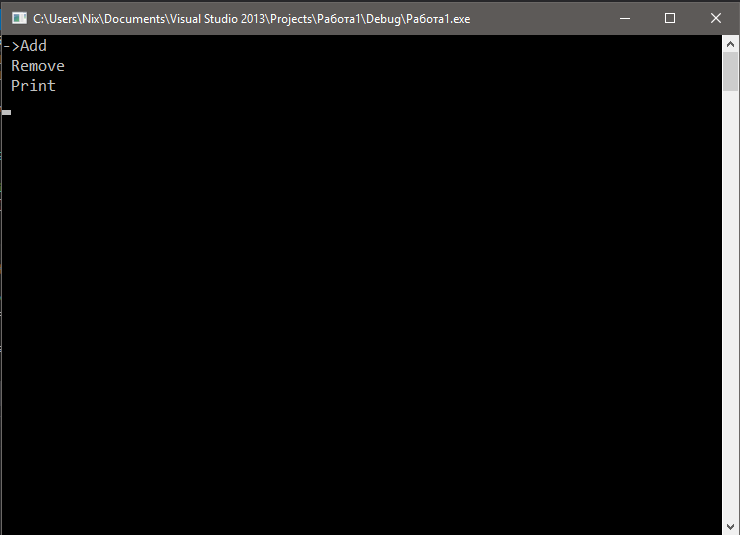
1. Делегаты.h

Содержит реализацию делегатов, которые используются в меню для хранения и динамического связывания объекта и метода. Реализация основана на шаблонах классов с несколькими специализациями.

1. Файл с реализацией функции main.

Для задания структуры меню используется initializer\_list<Items>. Добавление происходит в формате **Название пункта/объект/метод**, к примеру {"Add", &shop, &Shop::Add}. Добавление подменю в формате **Имя подменю/количество пунктов**, к примеру {"Settings", 3}, следующие три записи будут занесены в подменю.

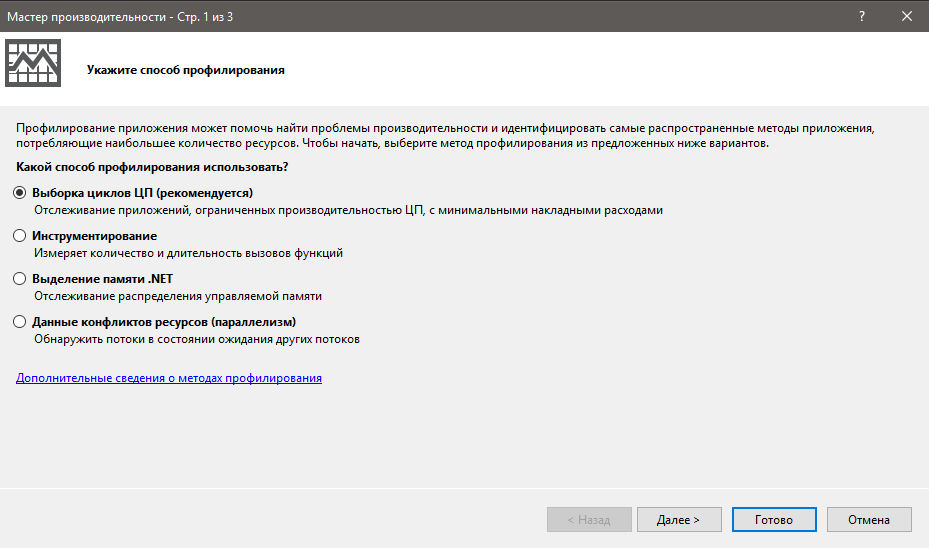
В итоге после сборки проекта и его запуска мы получим программу, которую в теории можно использовать в реальном магазине для ведения учета автомобилей. Скриншоты ее работы приведены ниже.



Запустим режим производительность и диагностика **Анализ- производительность и диагностика** (Alt+F4). В открывшемся окне не будем ничего менять и просто запустим приложение нажав **Начать**.

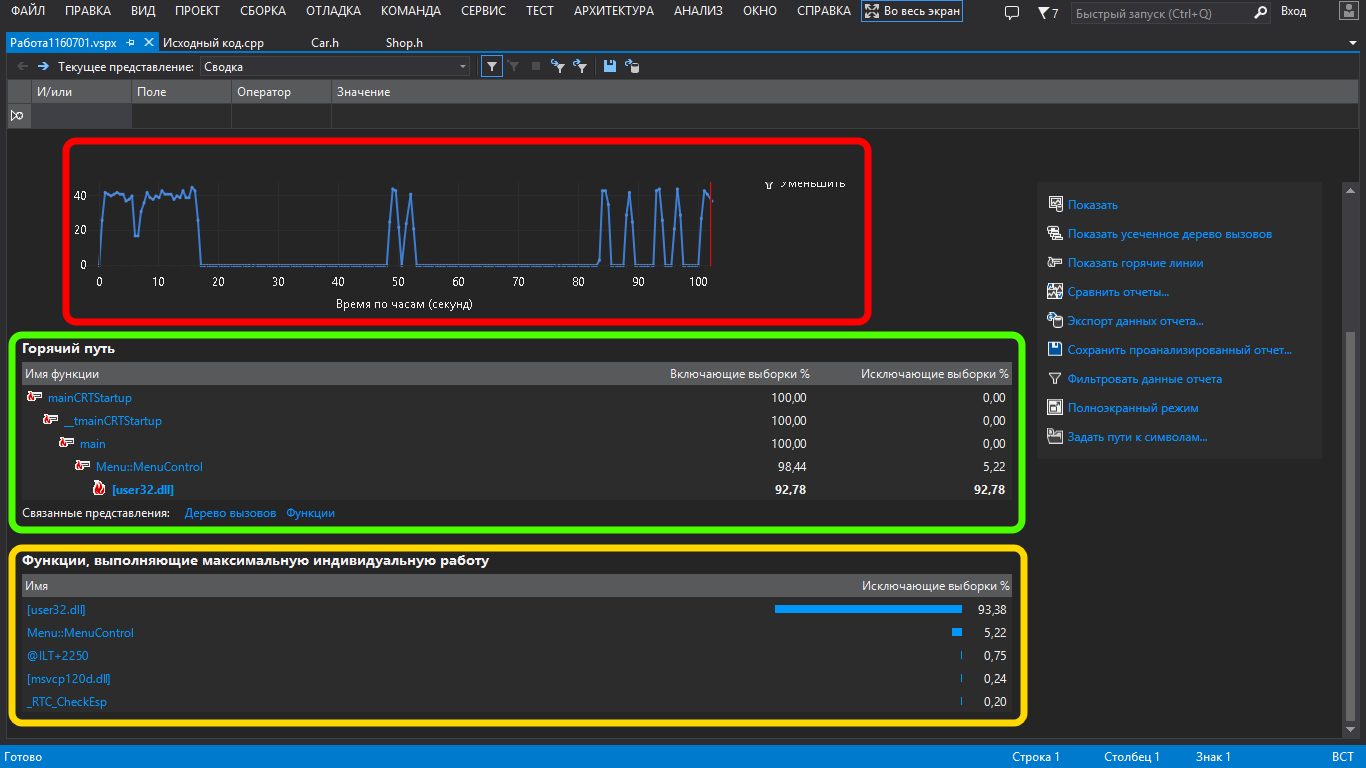
Открылось окно мастера производительности (Рисунок 9) в нем снова оставляем все как есть и жмем **Готово**.

Рисунок 9



Ждем запуска приложения, после завершения работы будет запущен режим профилирования (Рисунок 10).

Рисунок 10



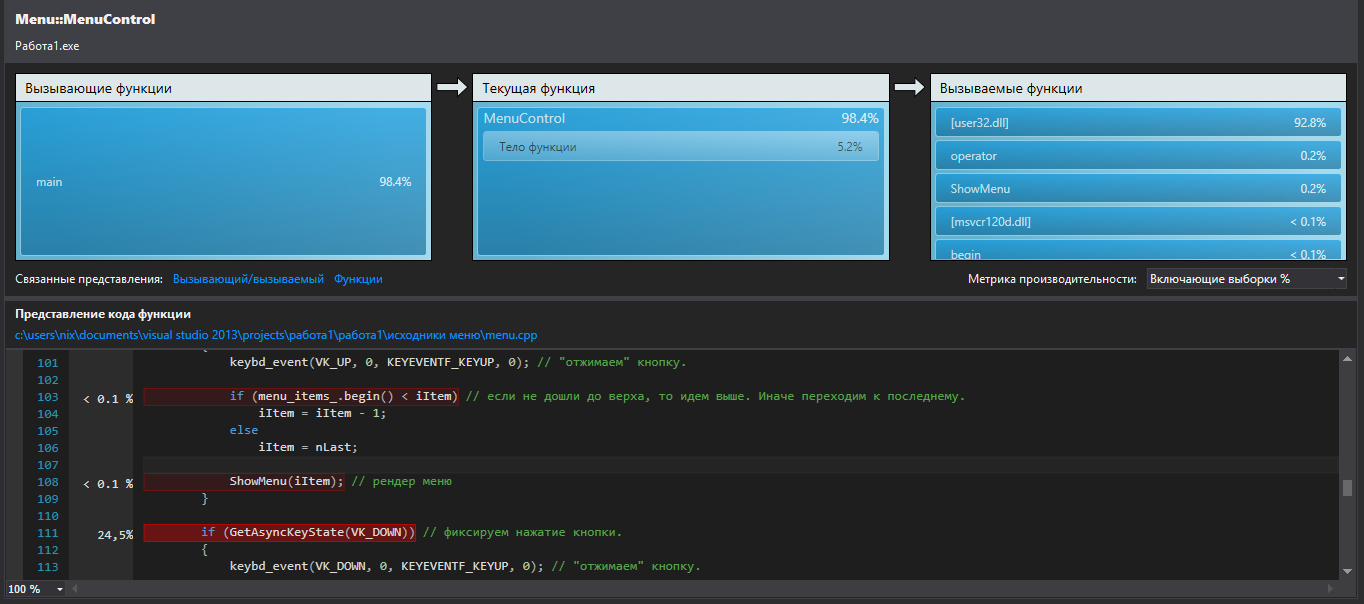
Красным выделен график загрузки ЦП в процессе работы.

Зеленым, стек вызовов функций.

Жёлтым, максимально задействованные функции.

Справа имеется список функций для работы с данными и их обработки. Интерес также представляет анализ функций в стеке (Рисунок 11)**.**

Рисунок 11

****

В нем можно перемещаться по стеку вперед и назад, узнавать какие функции вызывают текущую, а какие будут вызваны ей. Ниже показан код функции и какие её части работают и насколько загружают ЦП.

В этом разделе мы рассмотрели создание приложения на языке C++, в качестве интерфейса приложения использовали консоль ОС Windows. В процессе написания мы рассмотрели средства индикации ошибок синтаксиса, процесс их устранения и способы отладки приложения.

# Отличия Visual Studio 2013 от версии 2015 года

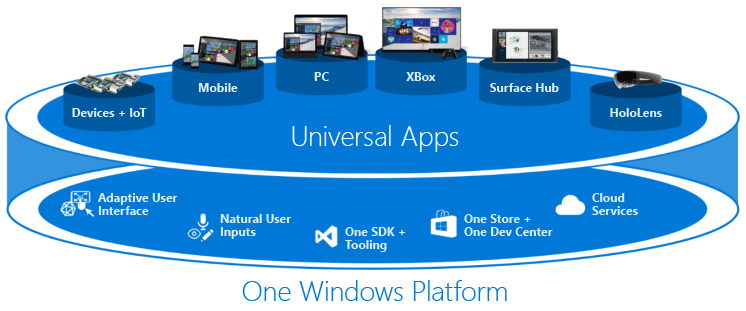
## Выбор целевой платформы.

Visual Studio 2015 поддерживает кроссплатформенную разработку для мобильных устройств. Внутри среды Visual Studio IDE можно программировать приложения и игры, ориентированные на iOS, Android и Windows, которые совместно используют общую базу кода. Все эти новые типы проектов можно будет видеть в диалоговом окне **Файл-новый проект**.

И конечно, обеспечивается еще более качественная поддержка для классических приложений с различными усовершенствованиями языков, библиотек и инструментов.

## Универсальные приложения для любого устройства на Windows 10.

Благодаря универсальной платформе Windows и единому ядру Windows одно и то же приложение можно запускать на любом устройстве Windows 10 — от телефонов и до настольных компьютеров. Используйте Visual Studio 2015 с универсальными средствами разработки приложений Windows для создания этих приложений.



## Интернет

Новая, созданная с нуля платформа разработки ASP.NET 5.

## .NET Framework

.NET Framework 4.6 Майкрософт предлагает около 150 новых API и 50 обновленных API для использования дополнительных сценариев. Например, все больше коллекций теперь реализуют IReadOnlyCollection<T>, что упрощает их использование. Кроме того, упомянутая ранее платформа ASP.NET 5 предлагает компактную платформу .NET для создания современных облачных приложений.

Приложения Магазина Windows, написанные на C# для платформы .NET Framework, теперь могут получить преимущество собственной платформы .NET, которая компилирует приложения с собственным кодом, а не с IL, а .NET Framework 4.6 также добавляет RyuJIT, 64-разрядный JIT-компилятор.

Новые компиляторы C# и VB ("Roslyn") значительно уменьшают время компоновки и предоставляют понятные API анализа кода. Visual Studio 2015 использует преимущество Roslyn для предоставления дополнительного рефакторинга, включая встроенное переименование, анализаторы и быстрые исправления.

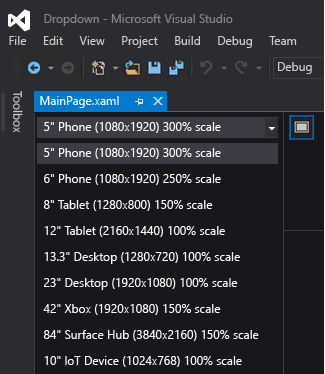
Языки C# и Visual Basic содержат множество небольших усовершенствований, которые касаются базового языка и поддержки IDE. Эти усовершенствования делают процесс написания кода .NET еще более интуитивным, удобным и продуктивным.

## C++

Visual C++ обеспечивает значительные усовершенствования в согласованности языка C++11/14, поддержку разработки кроссплатформенных приложений для мобильных устройств, поддержку возобновляемых функций и ожидания (в настоящее время запланировано для стандартизации в C++17), реализацию усовершенствований и исправлений ошибок в библиотеке времени выполнения C (CRT) и в стандартной библиотеке C++ (STL), возможность изменения размеров диалоговых окон в MFC, новый оптимизированный компилятор, улучшенную производительность сборки, новые возможности диагностики и новые инструменты производительности в редакторе кода.

## Строка меню предварительного просмотра устройств

В проектах универсальной платформы Windows строка меню предварительного просмотра устройств позволяет просматривать отрисовку пользовательского интерфейса на основе XAML на экранах различных размеров.

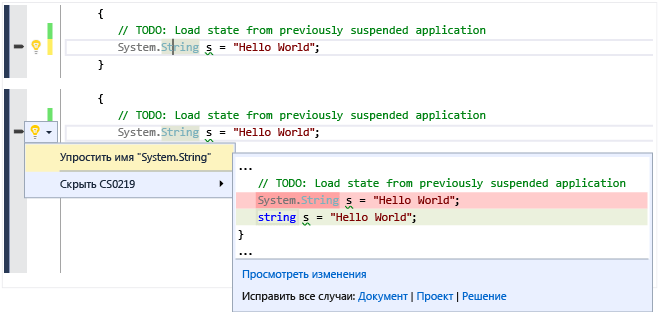


## Диагностика графики

С момента выпуска Visual Studio 2013 в инструменты диагностики графики Visual Studio было добавлено множество новых функций, включая анализ кадров, поддержку Windows Phone, редактирование и применение шейдеров и программы командной строки для захвата. Кроме того, была добавлена поддержка для отладки приложений DirectX12.

## Динамический анализ кода

Новый компилятор Roslyn для C# и Visual Basic не только быстрее компилирует, но также позволяет использовать полностью новые сценарии, такие как динамический анализ кода, который предоставляет подробный и настраиваемый отзыв и предложения непосредственно в редакторе кода по мере ввода данных. В Visual Studio 2015 лампочки отображаются в левой части (при использовании клавиатуры) или во всплывающей подсказке (при наведении указателя мыши на ошибку). Лампочка сообщает в режиме реального времени, что компилятор (возможно, использующий настраиваемый набор правил) обнаружил проблему в коде и предлагает вариант ее решения. Если вы видите лампочку, щелкните ее для получения предложений, требующих действий.



# Создание приложений с использованием WPF

## Описание WPF

**WPF (Windows Presentation Foundation**) — это платформа пользовательского интерфейса для создания клиентских приложений для настольных систем. Платформа разработки WPF поддерживает широкий набор компонентов для разработки приложений, включая модель приложения, ресурсы, элементы управления, графику, макет, привязки данных, документы и безопасность. Она является частью платформы .NET Framework. WPF использует расширяемый язык разметки для приложений (XAML), чтобы предоставить декларативную модель для программирования приложений.

Для начала разработаем визуальный интерфейс приложения с помощью встроенных средств редактирования Visual Studio. В интерфейсе нам понадобится две кнопки и поле вывода информации.

**Панель элементов** (Рисунок 12) содержит элементы управления которые можно разместить на макете приложения (Рисунок 13).

Рисунок 12

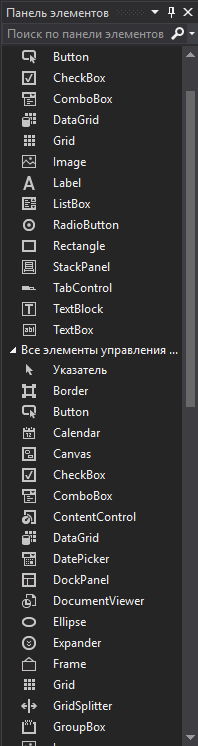
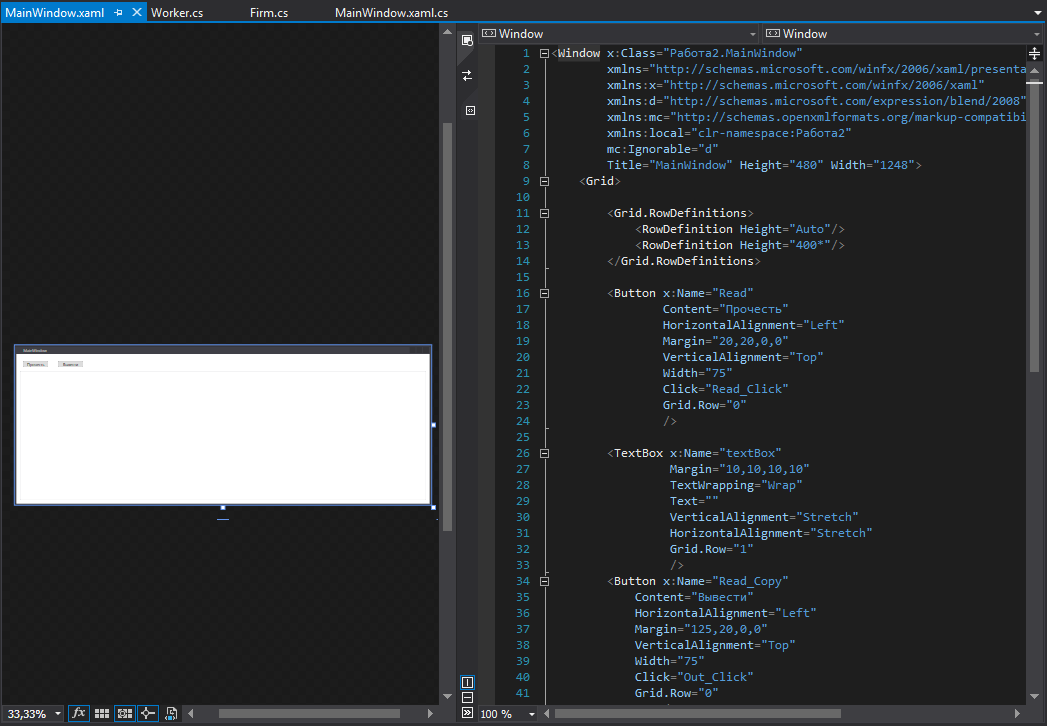


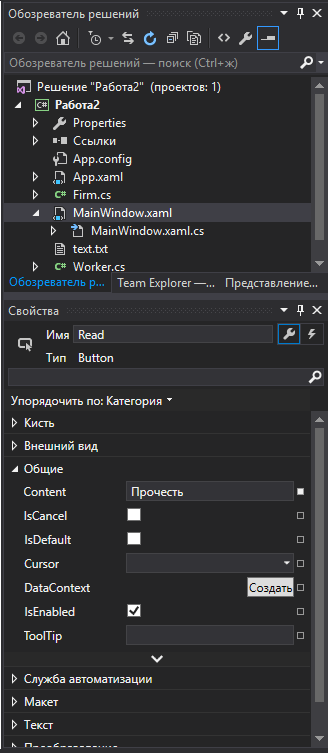
Рисунок 13



В редакторе справа можно вручную описывать свойства и положение объектов на форме, внесенные изменения немедленно будут на ней отображены. Панели можно переставлять между собой в любом удобном виде, на самой панели кода можно быстро перейти к нужному свойству или контейнеру размещения с помощью вкладок сверху. Также нужные свойства выбранного объекта можно настроить с помощью панели **Свойства** (Рисунок 14), еще в ней можно настроить ее назначение в программе или реакцию на действия пользователя.

На том же рисунке, в **Обозревателе решений**, можно заметить, что разметка макета программы связанна с кодом реализации логики работы конкретного окна. В этом файле описаны методы, реализующие реакцию программы на нажатие кнопок или наведения курсора на элемент.

Рисунок 14



## Написание приложения

Разработаем приложение «Учёт зарплаты работников». Пусть имеется фирма, в которой несколько отделов. Рабочий день длится 8 часов. Возможна переработка до 14 в сутки. Для каждого работника фирмы мы учтём фамилию, отдел, часы работы, стаж работы, зарплату в час, переработку, больничный по стажу. Также рассчитаем зарплату и штраф за прогулы.

В начале опишем класс Worker который будет представлять каждого работника.

Worker.cs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Worker

{

protected double experience;

protected double hospital;

protected double truancy;

protected double salary;

protected double hourwork;

protected bool overtime;

public double Experience{

get{ return experience; }

set{ experience += value+HourWork/(24+365); }

}

public double Hospital {

get{ return hospital; }

set{ hospital += value; }

}

public double Truancy {

get { return truancy; }

private set { truancy = (160-value>0 ? 160-value : 0); }

}

public double Salary {

get { return salary; }

private set { salary = value; }

}

public string Surname { get; set; }

public string Depart { get; set; }

public double HourWork {

get { return hourwork; }

set {

hourwork = value;

Truancy = value + hospital \* 8;

}

}

public double Rate { get; set; }

public void SalaryCalculate()=>Salary = ((hourwork<=160? hourwork \* Rate: 160\*Rate + (hourwork - 160) \*(1.5\*Rate))

- truancy \* (3 \* Rate) + (experience > 8 ? Hospital\*24 \* Rate : (Hospital\*8 \* (Rate/2))))\* 0.87;

public string PrintInfo()

{

string outstr = $"Фамилия:{Surname}\nОтдел:{Depart}\nЗарпалта:{salary}\n\n";

return outstr;

}

}

}

**SalaryCalculate** – метод который производит расчёт зарплаты, изменения суммы выплат обрабатываются непосредственно в выражение с помощью тернарной операции.

Так как в программе величины указаны в разных форматах, время работы и прогулы (часы), больничный (дни), стаж (годы), в свойствах и при вычислениях происходит конвертация. Учет пропусков идет по зачетному месяцу, то есть переработками можно закрыть необходимые часы. В качестве времени работы в месяц взято количество рабочих часов в месяц, 4недели\*5дней\*8часов, всего выходит 160 рабочих часов в месяц, переработки более 120 (4\*5\*14) часов в месяц учтены не будут.

Теперь опишем класс **Firm** который будет представлять собой конкретную организацию.

Firm.cs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class Firm

{

protected List<Worker> worker=new List<Worker> { };

public void Add(Worker worker\_) => worker.Add(worker\_);

public void PrintFirm(System.Windows.Controls.TextBox textBox)

{

foreach(Worker work in worker) {

textBox.Text+= work.PrintInfo();

}

}

}

В нем хранится динамический список для хранения работников фирмы и метод вывода всех работников компании в элемент интерфейса textbox, конкретный элемент будет передаваться в качестве параметра.

Класс MainWindow, как и его содержимое, является автоматически созданным, части, которые были написаны вручную будут описаны после.

MainWindow.xaml.cs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

public partial class MainWindow : Window

{

Firm university = new Firm();

public MainWindow(){

InitializeComponent();

}

private void Read\_Click(object sender, RoutedEventArgs e){

FileStream file1 = new FileStream(@"E:\Visual Studio 2015\text.txt", FileMode.Open);

StreamReader reader = new StreamReader(file1);

do

{

Worker worker = new Worker

{

Surname = reader.ReadLine(),

Depart = reader.ReadLine(),

Rate = Convert.ToDouble(reader.ReadLine()),

Experience = Convert.ToDouble(reader.ReadLine()),

Hospital = Convert.ToDouble(reader.ReadLine()),

HourWork = Convert.ToDouble(reader.ReadLine())

};

worker.SalaryCalculate();

university.Add(worker);

} while (!reader.EndOfStream);

reader.Close();

}

private void Out\_Click(object sender, RoutedEventArgs e) => university.PrintFirm(textBox);

}

Содержимое события **Read\_Click** было написано вручную. В нем мы производим открытие файла и читаем данные всех работников до тех пор, пока они не кончатся, каждый проход мы создаем новый объект заполняем его и заносим в объект класса **Firm**. события **Out\_Click** отвечает за вывод списка работников.

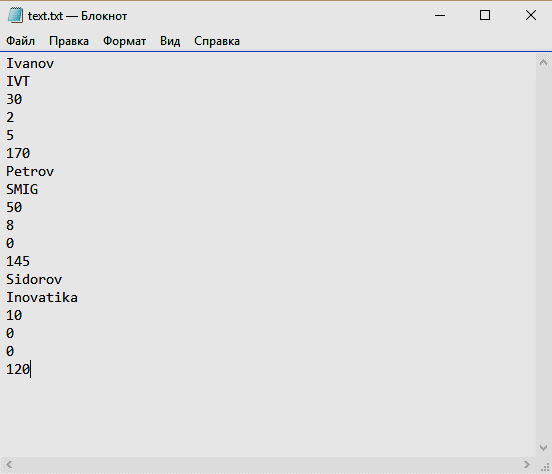
Для правильной работы программы данные в файле должны располагаться определённым образом:

1. Фамилия
2. Отдел
3. Ставка в час
4. Опыт работы (в годах)
5. Больничные (дней)
6. Часы работы за месяц

Отдельный группы данных не отделяются друг от друга и идут подряд.

## Пример работы приложения

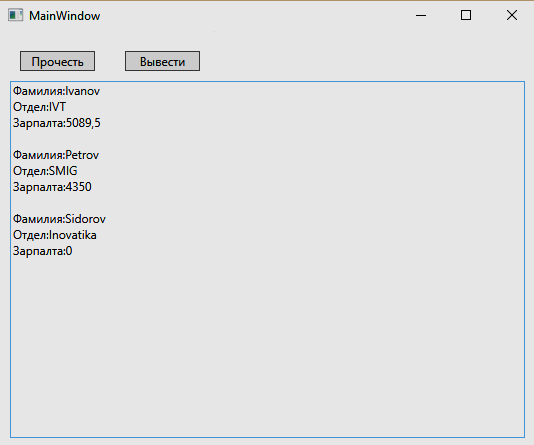
Рисунок 15



На рисунке выше представлен файл заполненный нужными данными.

Теперь запустим наше приложение и нажав кнопку «Прочитать» запустим подсчет зарплат.

Рисунок 16



Для верности можно провести расчеты вручную.

На этом создание приложения с использованием форм завершено.

# Список источников

1. msdn.microsoft.com/library
2. Wikipedia
3. C# 6.0. Справочник. Полное описание языка, Джозеф Албахари, Бен Албахари, Вильямс, 2016

1. Ошибки линковки корректно отображаются только на английском языке, также к ним нельзя сделать переход. [↑](#footnote-ref-1)
2. Удалить можно панели инструментов, созданные пользователем, но не панели инструментов по умолчанию. [↑](#footnote-ref-2)
3. Работает только для проектов на платформе .NET Framework [↑](#footnote-ref-3)
4. В основном используются «заголовочные» и файлы «исходного кода» [↑](#footnote-ref-4)
5. Файлы можно найти в приложение к методичке [↑](#footnote-ref-5)
6. Не более чем с одним аргументом [↑](#footnote-ref-6)