Министерство образования и науки Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования

«Санкт-Петербургский национальный исследовательский университет  
информационных технологий, механики и оптики»

Факультет информационных технологий и программирования

Кафедра информационных систем

Инструментальные средства разработки пО

Лабораторная работа № 1

Настройка Visual Studio Community 2017

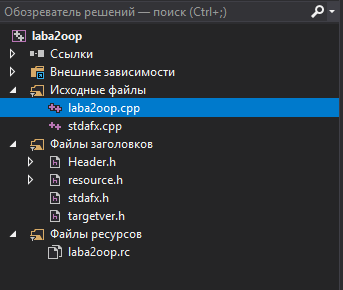
Выполнил студент группы M3306:  
Жаворонков С. В.

Проверил:  
Липкин Евгений Олегович

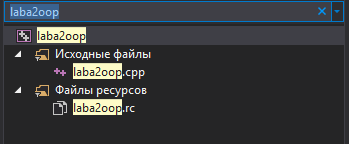
Навигация

На языке С++ в основном пишется код для серьёзных программ, и его размеры могут быть действительно огромными, из-за чего очень легко запутаться и потерять уйму времени. Продукт компании Microsoft предусмотрел для решения этой проблемы следующее.

В левой части окна располагается обозреватель решений



Он позволяет удобно и быстро работать с файлами: добавлять, удалять, изменять, а так же сворачивать и разворачивать. Так же стоит обратить внимание на то, что исходные файлы и заголовоные и ресурсные файлы располагаются отдельно друг от друга и не перемешиваются, из-за чего запутаться в них будет крайне сложно. Так же c помощью сочетания клавиш **Ctrl+;** осуществляется поиск по файлам, что ускоряет процесс работы, если в программе задействовано достаточно большое количество файлов



Сами файлы располагаются в верхней части текстового редактора, где их можно переставлять местами по своему усмотрению, или же закрывать ненужные.



Инструменты для навигации:

1. Поиск (**Ctrl+I**)

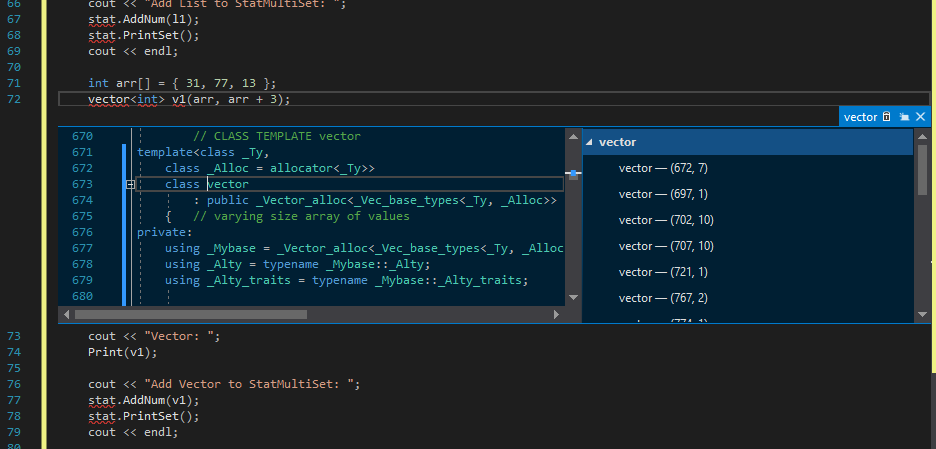
Это самый простой и достаточно быстрый способ найти строку в файле с кодом. Он полностью соответствует обычному поиску в текстовых файлах.

1. Поиск и замена (**Ctrl+F**)

Немногим отличается от «Поиска», но в отличие от него открывается в отдельном окне, а так же имеет несколько дополнений: учёт верхнего/нижнего регистра, возможность поиска отдельного слова.

1. Показать определение (**Alt+F12**)

С помощью этого инструмента можно посмотреть определение переменной, функции или класса Открывает окно в окне редактора текста, где указывается расположение выбраного объекта. Далее можно использовать инструмент «Перейти к реализации» (**Ctrl+F12**), он по сути выполнят обратное действие.



1. Перейти к определению (**F12**)

Аналогичный инструменту «Показать определение», но открывает небольшое окошко прямо в окне редактора текста, в котором показано место определения выбранного объекта.

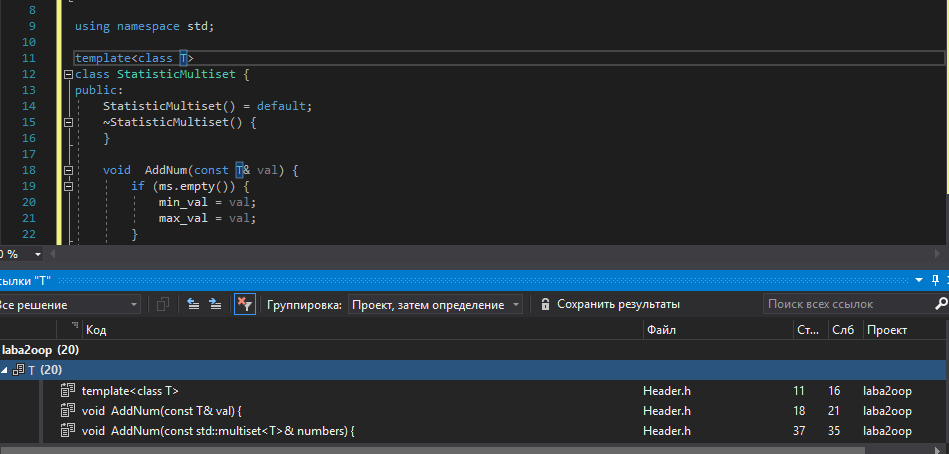


1. Посмотреть иерархию вызовов (**Ctrl+T**)

Применение этого инструмента на функцию открывает отдельное окно, в котором отображаются все вызовы выбранной функции в определённой области видимости (решение, проект, документ), все вызовы функций , которые вызывают её и так далее иерархично до самой первой вызывающей функции. В окне «Иерархия вызовов» можно перейти к определению любой упомянутой функци

1. Найти все ссылки (**Shift+F12**)

Элементом могут быть слова, которые не являются ключевыми, не находятся в комментариях, не являются строковым литералом: имя переменной, название класса, функция, тип, и т.д.. Этот инструмент подсвечивает все упоминания выбранного элемента в текущем файле и в отдельном окошке (при выборе в контекстном меню) упоминания слова во всем проекте или в выбранном файле.



Линтинг

Visual Studio сам отслеживает ошибки (но скорее даже опечатки) в коде, выделяя их аналогично текстовому редактору Word(подчёркивает красной волнистой линией слово, в котором допущена ошибка. К сожалению этот процесс пока не доведён разработчиками до совершенства.



Это работает следующим образом, средство анализа кода анализирует код с целью найти общие дефекты и нарушения рекомендаций программирования. Предупреждения, возникающие в ходе анализа кода, отличаются от ошибок и предупреждений компилятора, потому что средство анализа кода выполняет поиск конкретных шаблонов кода, которые могут вызвать проблемы при использовании и чтении кода. Анализ кода позволяет находить дефекты в коде, которые невозможно обнаружить в процессе тестирования.

Запускается анализатор кода либо с помощью горячих клавиш, либо через меню

«*Сборка* - > *Выполнить анализ кода в решении*». В свойствах проекта в меню «*Анализ кода*» можно включить автоматический запуск анализатора кода при сборке проекта и настроить правила которые будут использоваться при проверке.

Сборка

Процесс сборки запускается сочетанием клавиш **Ctrl+Shift+B**.

При создании проекта среда разработки Visual Studio создает для него конфигурации сборки по умолчанию, а также содержащее проект решение. Эти конфигурации определяют, как выполняется сборка и развертывание решений и проектов. В частности, используются уникальные конфигурации проектов для разных целевых платформ (например, Windows или Linux) и типов сборки (например, отладка или выпуск). Разработчик может как угодно изменять эти конфигурации и при необходимости создавать свои собственные.

Процесс сборки состоит из трансляции кода и его компоновки

1. Трансляция — это общее название для преобразования текста программы (из кода, написанного на высокоуровневом языке программирования, в нашем случае С/С++).

Подразделяется на виды:

* 1. **Компиляция.**  
     Компиляция выполняется один раз и во время неё исходный код программы преобразуется в машинный код. Производительность программы достигается, помимо всего прочего, работой компилятора, оптимизирующего исходный код.
  2. **Интерпретация.**

**Чистая интерпретация.**  
Интерпретатор читает и сразу исполняет код. В этом случае не происходит преобразование из одного языка в другой.

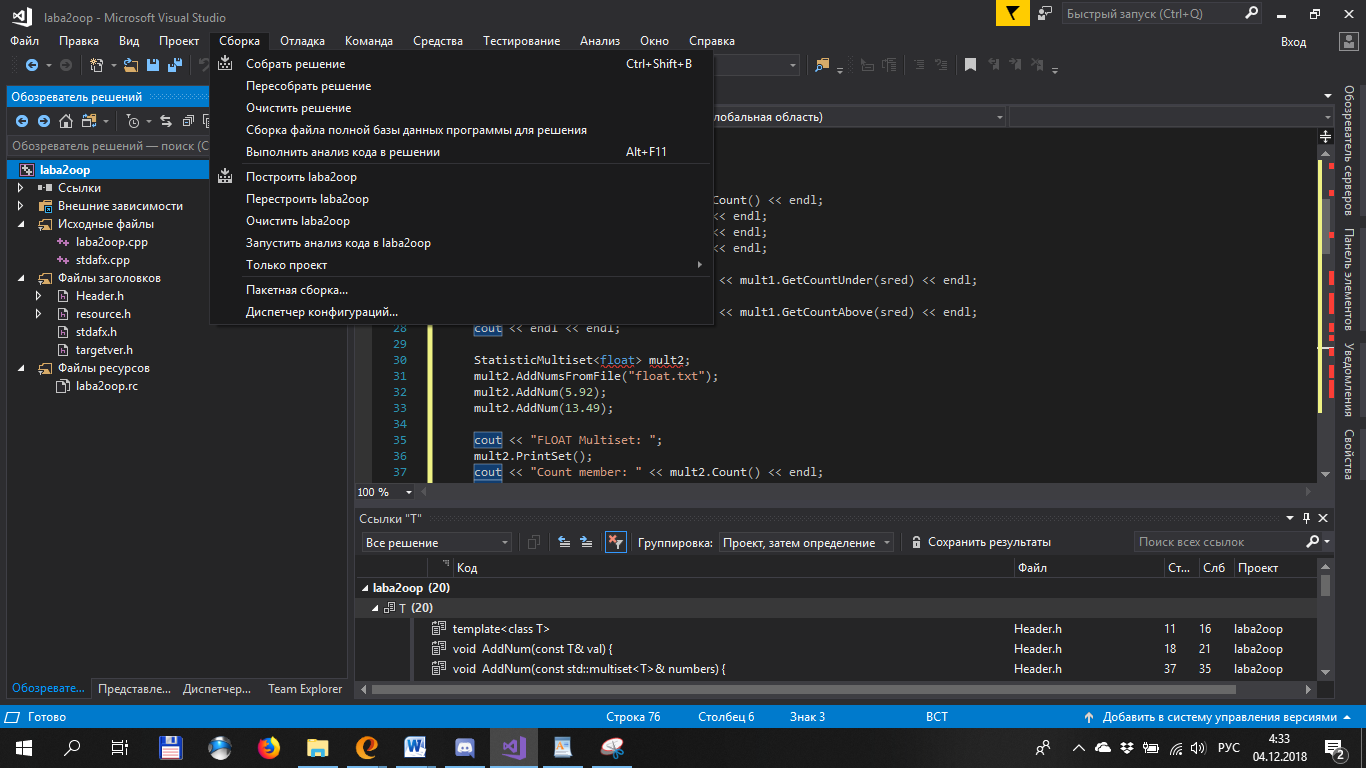
**Смешанная реализация.**  
Интерпретатор читает код, создаёт его промежуточное представление в памяти, например, в виде байт-кода, и затем выполняет созданное промежуточное представление.

**Динамическая компиляция (JIT-компиляция).**  
Компиляция в этом случае выполняется много раз, но для разных участков кода. Во время первого обращения к не скомпилированной части программы (функции, например) происходит её компиляция и сохранение скомпилированного кода в памяти. При повторном обращении к этому же участку используется уже созданный машинный код.

1. Процесс компоновки (линковки) заключается в объединении всех объектных файлов проекта, созданных на этапе трансляции, в один исполняемый файл или библиотеку.

Во время сборки создаётся манифест сборки, который позволяет системе определить все файлы, входящие в сборку, сопоставить ссылки на типы, ресурсы, сборки с их файлами, управлять контролем версий. Если у сборки отсутствует манифест, то заключённый в ней код выполняться не будет.

Операции, связанные со сборкой, находятся в меню «Сборка» на верхней панели. Настроить параметры можно в свойствах проекта в пункте «Сборка».



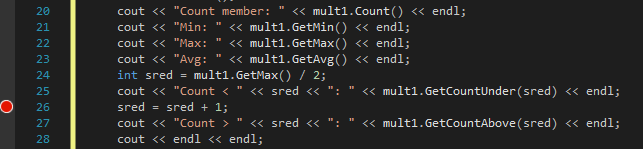
Тестирование

* Обозреватель тестов позволяет легко интегрировать модульные тесты в вашу практику разработки. Можно использовать платформу для выполнения модульных тестов Microsoft или одну из нескольких сторонних платформ, в том числе платформы с открытым исходным кодом.
* IntelliTest автоматически создает модульные тесты и тестовые данные для управляемого кода.
* Объем протестированного кода позволяет определить, какая часть кода проекта в действительности тестируется закодированными тестами, такими как модульные тесты.
* Microsoft Fakes помогает изолировать тестируемый код, заменяя другие части приложения заглушками или оболочками.
* Функция Live Unit Testing автоматически выполняет модульные тесты в фоновом режиме и графически отображает объем протестированного кода и результаты тестирования в редакторе кода Visual Studio.
* С помощью закодированных тестов пользовательского интерфейса можно проверять работу пользовательского интерфейса приложения.
* Нагрузочное тестирование позволяет моделировать нагрузку на серверное приложение посредством выполнения модульных тестов и веб-тестов производительности.

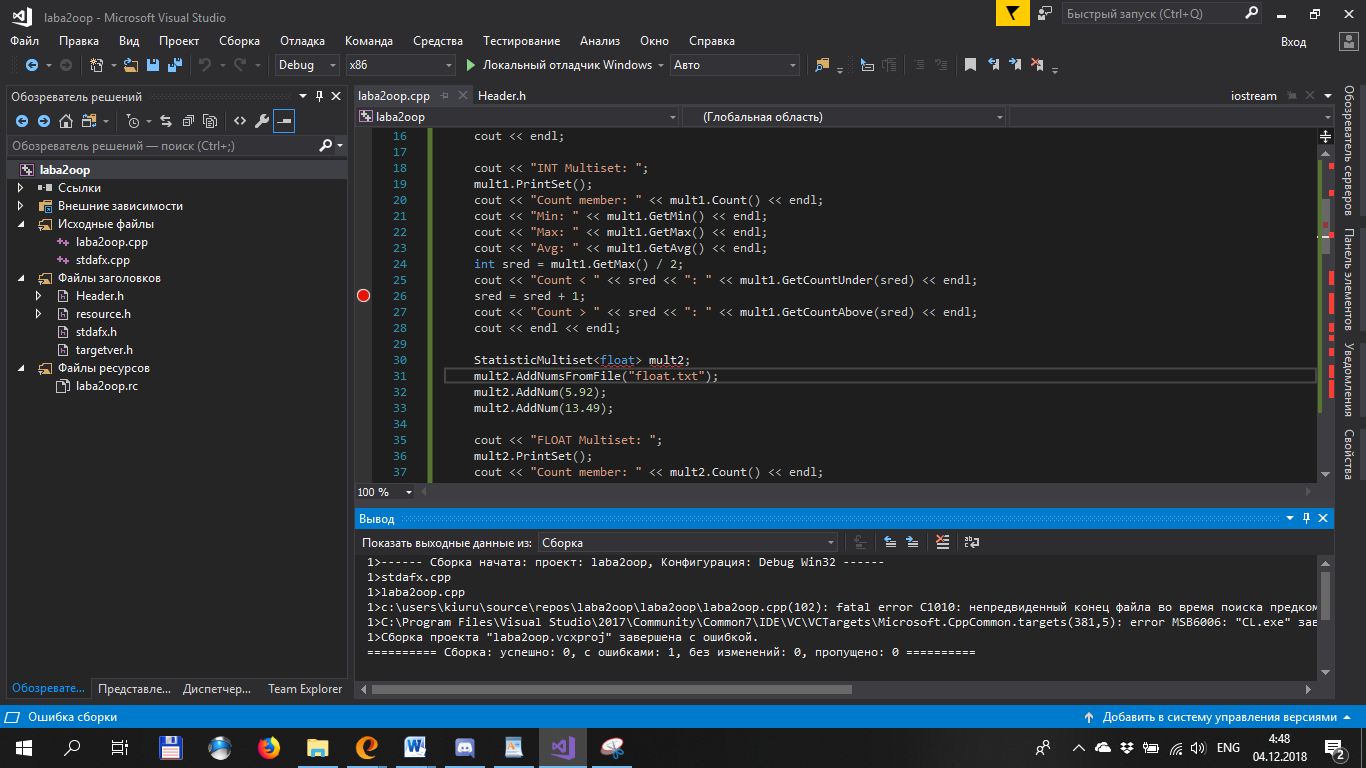
Отладка

Отладчик предоставляет множество способов, чтобы увидеть, что делает код во время его выполнения. Можно пошагово выполнять код и просматривать значения, хранящиеся в переменных, можно задать Контрольные значения для переменных, чтобы увидеть при изменении значений, можно просмотреть путь выполнения кода, и т.п

Так можно установить точку остановки запустить отладчик. Слева от строки нужно нажать на столбец и появиться красная точка

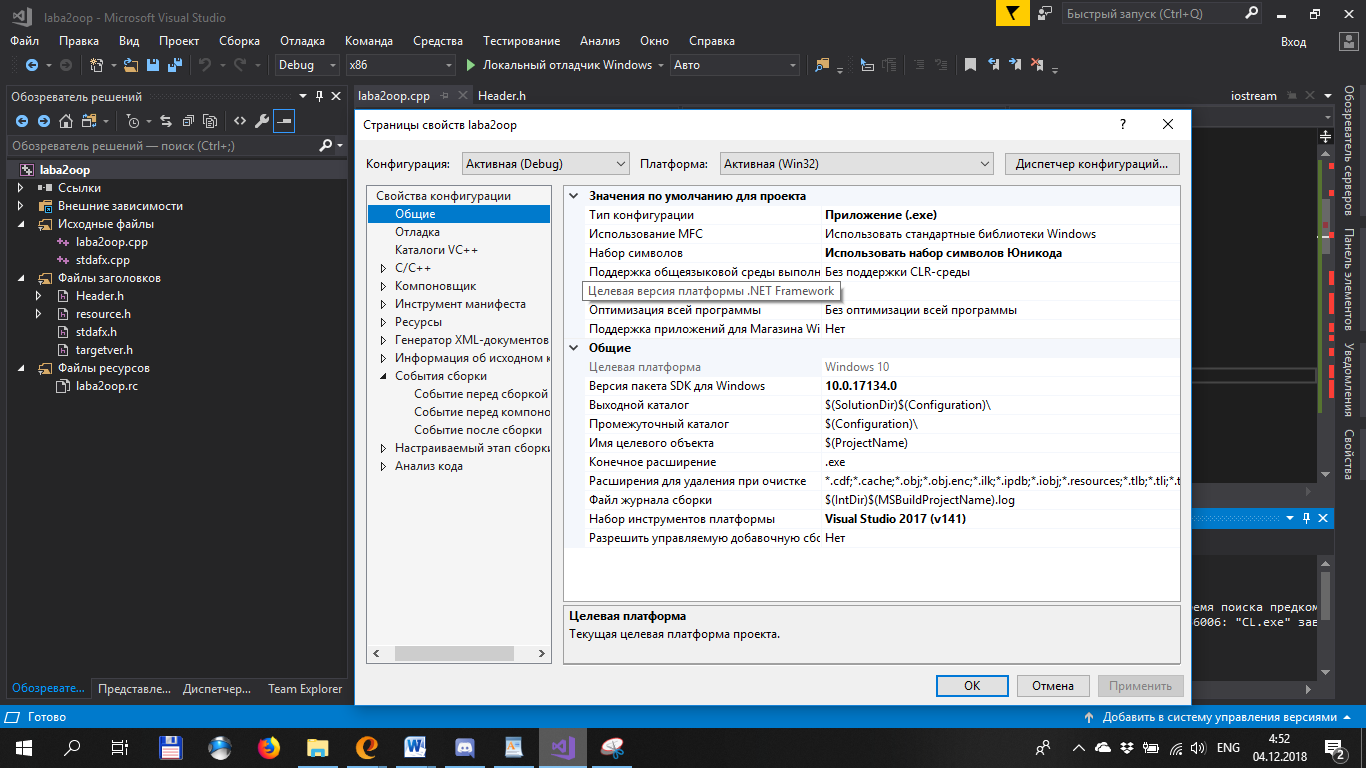


Процесс отладки запускается клавишей **F5**



Скрипты

Visual studio так же предоставляет возможность создания скриптов в виде, например, BAT файлов, указывающих последовательность действий до и/или после сборки. Располагаются данные настройки по пути *Проект* - > *Свойства* - > *События построения*.



Горячие клавиши

Горячие клавиши ускоряют разработку приложений, позволяя быстрее получать доступ к различным инструментам Visual Studio. Причём это относится не только к программированию, а к любой работе за компьютером, поэтому знание основных часто используемых сочетаний клавиш может быть очень полезно для повышения эффективности и скорости работы.

Вот список некоторых горячих клавиш, которые чаще всего используются мною и которые я собираюсь запомнить (запятая между комбинациями обозначает, что их нужно нажать последовательно):

* Комментирование.

Есть несколько комбинаций клавиш, для создания комментирования:

Ctrl+E,C — закоментировать выбранные линии в коде.

Ctrl+E,U — раскомментировать выбранные линии в коде.

Если перед имененм метода набрать /// — автоматически будут добавлены XML комментарии, которые потом можно использовать для создания документации.

* Поиск.

Ctrl+F — открывается стандартное окно поиска, если в буфер недавно было скопирован текст, то он автоматически подставиться.

Ctrl+F3 — поиск «активного» текста вниз (который выделен или слово на котором стоит курсор), без копирования в буфер обмена.

Ctrl+Shift+F3 — поиск «активного» текста вверх.

Alt+Enter - выберите все вхождения поиска.

Ctrl+D - добавить выделение в следующий, найти совпадение

Ctrl+K - перенести последний выбор на следующий

Alt+C / R / W - переключить регистр-регистр / regex / целое слово Форматирование и редактирование

Ctrl+Shift+V — не отпуская Ctrl+Shift и нажимая на V, вы сможете вставить один из 10 последних кусков текста, скопированных в буфер обмена.

Tab — увеличить отступ. Shift+Tab — уменьшить отступ.

* Навигация

Ctrl+T - Показать все символы

Ctrl+G - Перейти к строке …

Ctrl+P - Перейти к файлу …

Ctrl+Shift+O - Перейдите к символу …

Ctrl+Shift+M - Показать панель задач

F8 -Перейти к следующей ошибке или предупреждению

Shift+F8 - Перейти к предыдущей ошибке или предупреждению

Ctrl+Shift+Tab - Навигация по истории группы редакторов

Alt+ ← / → - Вернуться назад / вперед

Ctrl+M - Переключить вкладку перемещения фокуса Общие

Ctrl+Shift+P, F1 - Показать палитру команд

Ctrl+P - Быстрое открытие, переход в файл …

Ctrl+Shift+N - Новое окно / экземпляр

Ctrl+Shift+W - Закрыть окно / экземпляр

Ctrl+, - Пользовательские настройки

Ctrl+K Ctrl+S - Горячие клавиши Управление файлами

Ctrl+N - Новый файл

Ctrl+O - Открыть файл…

Ctrl+S -Сохранить

Ctrl+Shift+S -Сохранить как…

Ctrl+K S - Сохранить все

Ctrl+F4 - Закрыть

Ctrl+K Ctrl+W - Закрыть все

Ctrl+Shift+T - Повторно открыть закрытый редактор

Ctrl+K Enter - Сохранить редактор режима предварительного просмотра

Ctrl+Tab - Открыть следующий

Ctrl+Shift+Tab - Открыть предыдущий

Ctrl+K P - Копировать путь к активному файлу

Ctrl+K R - Показать активный файл в проводнике

Ctrl+K O - Показать активный файл в новом окне / экземпляре Отлаживать код (debug)

F9 - Переключить точку останова

F5 - Пуск / Продолжить

Shift+F5 - Стоп

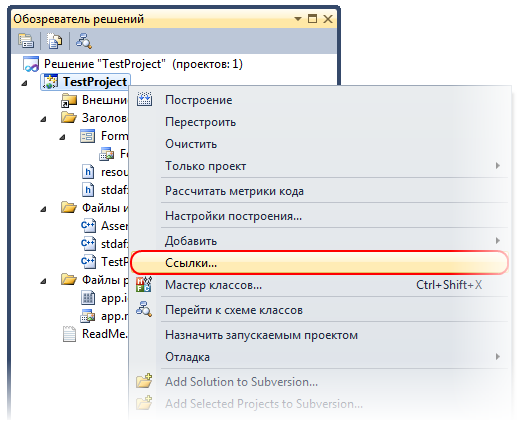
F11 / Shift+F11 - Ввод / выключение

F10 - Переступить

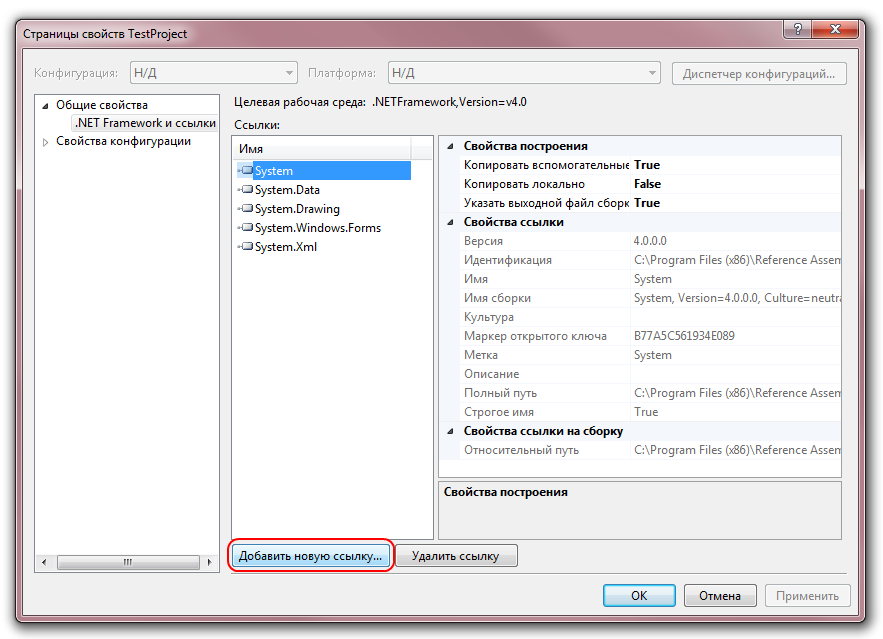
Ctrl+K Ctrl+I - Показать зависание

### Подключение библиотеки OsmModbusControl.dll к CLR-проекту Visual Studio на Visual C++

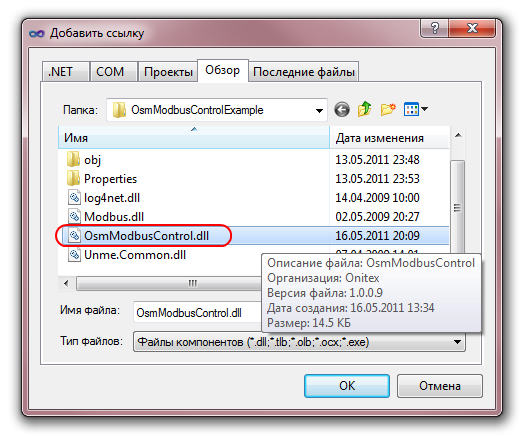
Для того, чтобы подключить библиотеку к проекту, необходимо сделать следующее:  
1. Перенести папку OsmModbusControl.dll из архива с SDK в папку проекта (желательно).  
2. Открыть в Visual Studio свой проект (или создать новый), и добавить ссылку на файл OsmModbusControl.dll через Обозреватель решений (Solution Explorer). Для этого необходимо кликнуть правой кнопкой мыши по названию проекта и выбрать в меню пункт Ссылки (References):



3. В появившимся окне необходимо нажать кнопку Добавить новую ссылку... (Add New Reference...):



4. В появившимся окне необходимо перейти на вкладку Обзор (Browse), перейти в папку OsmModbusControl.dll в папке проекта, выбрать файл OsmModbusControl.dll и нажать Ок. Все остальные, необходимые для работы библиотеки, будут добавлены в проект автоматически.



5. Прописать в заголовочном файле вашего проекта ссылку на пространство имен OsmModbusControl , после чего можно использовать классы и методы библиотеки, как это показано в следующем примере консольного приложения (использование библиотеки в визуальных приложениях (Windows Forms) аналогично, для более подробной информации см. пример использования и документацию):

// Test.cpp: пример простейшего консольного приложения.  
#include "stdafx.h"  
   
using namespace System;  
using namespace OsmModbusControl; // Подключение пространства имен.

В разделе свойств C/C ++ свойств проекта на вкладке Общие. Добавьте туда путь к вашей недавно созданной папке include . Это позволит вам включить заголовочный файл и заставить его работать правильно.

В разделе свойств Linker свойств проекта, также на вкладке Общие имеется соответствующая запись для Дополнительные каталоги библиотек . Добавьте туда путь к папке lib . На следующей вкладке: Вход есть запись для Дополнительные зависимости . Добавьте там фактическое имя файла библиотеки.

Эти шаги должны позволить, чтобы ваш проект был построен с использованием файлов .h , .lib и .dll .

