# Создание консольного приложения в VS2010

- Запускаем MS Visual Studio.
- "New project" на "Start page" (или "File | New — Project").
- "Project types" выбрать "Visual C++ —Win32".
- "Templates" выбрать "Win32 Console Application".
- В "Name" указать имя, в "Location" путь к папке проектов. Папка с именем, совпадающим с именем проекта, создастся автоматически. "OK".
- "Next". Убеждаемся, что выбрано консольное приложение. Здесь я советую снять галочку с "Precompiled headers" (Предкомпилированные заголовки), в нашем случае от них проблем больше, чем пользы. "Finish".
- Видим следующий код:

```
// TestProject.cpp : Defines the entry point for the console application.
//

#include "stdafx.h"


int _tmain(int argc, _TCHAR* argv[])
{
        return 0;
}
```

- Удаляем его целиком и пишем:

```
#include <cstdio>

int main(int argc, char* argv[]) {
  printf("Hello, world!\n");
  return 0;
}
```

Сразу рекомендую в "Project | Properties — Configuration Properties — General" поменять "Character set" с "Unicode" на "Multi-Byte" (также в свойства проекта можно быстрее попасть, нажав <Alt + F7>).

- Нажмем <Ctrl + F5> или "Debug | Start without debugging" или кнопку "Start without debugging" (красный восклицательный знак) — "OK".
  Чтобы запускать приложение с отладкой, нужно запускать программу по <F5> или "Debug | Start debugging" или по кнопке "Start" (треугольничек) (это три эквивалентных варианта). Но тогда консольное окошко сразу закрывается. В этом случае можно пользоваться точками остановки (щелчок слева от `return 0;` на сером фоне; например). Запуск по <F5> выполняет программу до ближайшей точки остановки или до конца программы. Другой вариант - добавить последней строчкой `system("pause");` включив в начале `#include <cstdlib>`.
- Если какой-то часто используемой кнопки на панели нет, то ее можно добавить с помощью "Tools | Customize". Там в разделе с названием таким же, как пункт меню (например, "Debug"), нужно найти нужную кнопку и перетащить на панель горячих кнопок.
- До сих пор, по умолчанию, компиляция шла в режиме "Debug". Этот режим необходим для отладки, но имеет недостатки: программа выполняется существенно медленнее, часть ошибок может не проявиться и выполняемый exe-файл больше по размеру. Поэтому, когда программа отлажена, запускать ее лучше в "Release". Перейти в "Release" можно через раскрывающееся окошко вверху (там, где "Debug") или через "Build | Configuration Manager (Active Solution Configuration)".

**Внимание:** при попытке отладки с отслеживанием значений переменных в режиме "Release" могут выдаваться бессмысленные значения, поэтому в режиме "Release" отлаживать программу нельзя.

## Свойства проекта

Обратите внимание, что для "Release" и "Debug" опции разные, поэтому изменения (при необходимости) нужно вносить два раза.

Войти в редактирование свойств можно через "Project | Properties"

Изменения могут потребоваться в следующих моментах.

1. При создании проекта мы убрали галочку относительно использования precompiled headers (предкомпилированных заголовков). Система с precompiled headers позволяет убыстрить компиляцию, так как стандартные файлы компилируются один раз и результат сохраняется на диске. В этой системе при подключении стандартных файлов через `#include` их нужно (чтобы получить ускорение) подключать внутри файла `stdafx.h`. Если файлы подключаются не внутри, а в самом файле .cpp, то это нужно делать после строчки `#include "stdafx.h"`. Система с precompiled headers подключается/отключается в "Configuration properties | C/C++ (раскрыть) — Precompiled Headers (Use… или Not using…)". Если выбрано "Use…", то в любом из включаемых в проект файлов первой должна быть строчка `#include "stdafx.h"`. Хотя для конкретного файла можно и отключить использование предкомпиляции. Например, если вы включаете в проект стандартный файл без этой строчки, то можно сделать следующее: в Solution Explorer выделяете нужный файл и по контекстному меню, вызываемому по правой кнопке, выбираете Properties и уже там выбираете "Not Using…". Вообще, в консольном приложении предкомпиляция заголовков не нужна. А вот в приложении под Windows — уже гораздо более существенна.
2. Иногда существенным бывает выбор используемой кодировки. По умолчанию в Properties установлено "Configuration Properties — General — Character Set — Unicode". Если не делать специальных действий, направленных на использование Unicode, то лучше поменять "Character set" на "Multi-Byte".

## Редактор

Настройки редактора (лучше сделать сразу):

1. "Tools | Options", в списке выбираем "Text Editor — C/C++ — General", справа в разделе "Display" ставим галочку "Line numbers". Слева, чуть ниже, "Tabs", справа переключаем на "Insert spaces", "Indenting" на "Smart", устанавливаем "Tab size" и "Ident size" по вкусу (я настоятельно рекомендую 2).

Удобные приемы:

1. Выделение строки с помощью клика мышью слева и выделение слова двойным кликом.
2. Копировать, вырезать и вставлять можно стандартными горячими клавишами <Ctrl + C>, <Ctrl + X>, <Ctrl + V>. При вырезании/копировании ничего не выделено, то вырезается/копируется вся строчка под курсором.
3. Вставить/убрать комментарии можно с помощью кнопочек с нарисованными строчками (или "Edit | Advanced — Comment/Uncomment selection". Выделять лучше строки целиком, проведя мышкой с нажатой левой кнопкой по левой границе текста. Тогда комментарий вставится построчный.
4. Текст должен быть отформатирован. Руками вставлять пробелы не надо. Отступы или делаются табуляцией, автоматически, или выделенный текст можно отформатировать с помощью "Edit | Advanced — Format Selection" (или соответствующее сочетание клавиш).
5. Добавлять/убирать файлы нужно через Solution Explorer, а «ходить» по программе — с помощью Class View. Если какого-то окна нет, его можно добавить с помощью "View | …" или, "View | Other windows — …".
6. Удобная функция – перейти к объявлению или к определению объекта. Правой кнопкой на объекте и выбираем "Go to definition/Go to declaration". Еще более удобно навести на объект мышку, нажать <Ctrl> и щелкнуть по получившейся ссылке, сразу перейдем к определению.
7. <Ctrl + Space> - автодополнение. Дополняет идентификаторы и поля объектов. Умное, но не дополняет ключевые слова.

## Отладка

Отлаживать программу необходимо цивилизованным путем. А именно, с помощью Debugger (отладчика), позволяющего прерывать (замораживать) исполнение программы и исследовать ее состояние (например, просматривать значения переменных).

Стандартный инструмент отладки — точка остановки (Breakpoint). Поставить/снять точку остановки — клик мышкой по серой полоске слева от строчки или <F9>. <F5> (или кнопка с треугольничком) — запуск программы до конца или до следующей точки остановки. Есть окошко Breakpoints, с помощью которого можно ими управлять (если нет, то "Debug | Windows — Breakpoints") например, отключать, устанавливать Condition (условие, при котором точка сработает) или Hit Count (после какого по счету прохода точка должна сработать).

Также бывает полезна пошаговая отладка. Для нее есть панель Debug (если нет, то "View | Toolbars — Debug") и там нужно использовать Step over (<F10>, переход на следующую строчку без захода внутрь функции), "Step in" (<F11>, с заходом), "Step out" (<Shift + F11>, выход из функции, в которую зашли). Пошаговую отладку можно чередовать с передвижением по breakpoints.

Чтобы просматривать и менять (sic!) значения переменных есть окошко Watches, точнее, несколько окошек. Autos, Locals, Watch1-Watch4 ("Debug | Windows — Autos, Locals, Watch"). Отображаются соответственно задействованные в текущем блоке переменные, локальные переменные и еще четыре окошка можно наполнить любыми интересующими переменными. Если имя переменной не имеет смысла в текущей точке выполнения, Вам об этом так и напишут. Еще можно смотреть значения переменных, просто наводя курсор мыши на них в редакторе. Если выделить какое-то выражение и навести курсор, то выражение будет вычислено и будет отображено значение.

С помощью окошка Quick Watch ("Debug | Quick Watch" или <Shift + F9>) можно посмотреть значение переменной или выражения.

Еще есть полезное окошко "Call Stack" (Стек вызовов), если нет, то "Debug | Windows — Call Stack". В нем можно смотреть всю последовательность вызовов функций (так называемый Stack Trace). Также можно "развернуть стек", т.е. переместиться в то место, из которого был совершен конкретный вызов функции в момент непосредственно перед вызовом и изучить контекст (т.е. состояние локальных переменных на момент вызова).

Если программа запускается в режиме отладки, то при сбое автоматически запустится отладчик. По Call Stack обычно можно определить последовательность событий перед ошибкой и, вернувшись назад по вызовам, определить, что к ошибке привело.

Если надоело отлаживать или вы уже поняли, где ошибка, то выйти из Debugger’а можно по клавишам <Shift + F5> или "Debug | Stop Debugging".

## Работа с проектом

Начнем с того, как добавить файл в пустой проект. Для того чтобы создать проект с пустым консольным приложением, делаем все то же самое, но заходим на страницу Application Settings и там ставим галочку "Empty Project". Заметим, что при таком способе создания проекта по умолчанию не устанавливается опция "Use precompiled headers".

Чтобы добавить файл, нужно:

- Зайти в "Solution explorer", выделить имя проекта и по правой кнопке мышки выбрать "Add | Add new Item" В окне "Add new Item — Templates — Visual C++ — Code — File C++(.cpp)"
- Указать имя в "Name".
- "Open"

Если файл уже создан, то выбираете "Add | Add Existing Item". При этом можно одновременно выделить несколько файлов (используя <Ctrl> и <Shift>), и .cpp, и .h, – они попадут в нужные папки проекта.
Папки проекта не обязательно отражают реальную структуру каталогов на диске. Можно добавлять файлы в проект из самых разных мест. Но для .h файлов существенно, чтобы они лежали в каталоге проекта. .cpp-файлы можно добавлять откуда угодно, однако лучше держать все исходные коды ФИЗИЧЕСКИ в одном каталоге.

**Замечание.** Solution (.sln) – это набор проектов/задач (.vcxproj), т.е. в один Solution можно добавить несколько проектов. Компилировать/строить (build) можно каждый проект по отдельности, а можно всё solution вместе (т.е. все входящие в него проекты). Если в Solution имеется несколько проектов, то запускаться, собираться и отлаживаться по умолчанию будет активный проект. Чтобы назначить проект активным, нажмите на него правой кнопкой мыши (в Solution Explorer) и выберите "Set as a startup project".

**Внимание:** после изменения опций или если какая-то непонятная ошибка, то всегда полезно сначала сделать "Build | Rebuild", так как не всегда корректно отслеживается, что нужно перекомпилировать, а что — нет.

## Необходимые файлы

Для переноса файлов с компьютера на компьютер нужен следующий набор файлов: *.cpp, *.h, *.vcproj, *.sln (если программа под Windows, то еще и *.rc и директория /res/ с ресурсами). Убедительная просьба не копировать все файлы, потому что там есть база данных IntelliSense, размером в десятки мегабайт. Кстати, из-за привычки VisualStudio постоянно эту базу перезаписывать, работа с проектом на флешке или сетевом диске может быть очень медленной.