Создание консольного приложения в VC*2010*

- 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. Галочки Create directory for solution НЕ должно быть. OK.
- Можно перед Finish нажать Next (или выбрать слева Application settings) и увидеть, что выбрано консольное приложение и стоит галочка на Precompiled header.
- "Finish"
- Видим следующий код:

```
#include "stdafx.h"

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

- Перед `return` добавляем строчки

```
	int group;
	printf("Input the group number please ");
	scanf("%d",&group);
	printf("You are from %3d\n", group);
```

- Нажмем "<Ctrl|F5>" или Debug|Start without debugging или кнопку Start without debugging (красный восклицательный знак) - "OK"
- Нестандартный вид _tmain и _TCHAR вызван тем, чтобы была совместимость с Unicode. Пока не будем с этим заморачиваться и заменим на

```
int main(int argc, char* argv[])
```

Тоже должно компилироваться. Если не компилируется, то тогда в Project – Properties - Configuration Properties – General нужно поменять Character set с Unicode на Multi-Byte.

- 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"); и включить #include <stdlib.h>. Например, так:

```
#include "stdafx.h"
#include <stdlib.h>

int main(int argc, char* argv[])
{
   int group;
   printf("Input the group number please ");
   scanf("%d",&group);
   printf("You are from %3d\n", group);
   printf("Press any key\n", group);
   system("pause");
   return 0;
}
```

- 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, тем более, что в режиме Release могут проявляться ошибки, которые были скрыты в Debug. 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

Во-первых, нужно заметить, 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 была возможность убрать галочку относительно использования precompiled headers (прекомпилированных заголовков). Мы ее не убрали, поэтому в проекте появилась строчка `#include "stdafx.h"`, а в проект добавились два файла – stdafx.h и stdafx.cpp (см. окошко с Solution Explorer). 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 должна быть строчка #include "stdafx.h". Хотя для конкретного файла можно и отключить использование прекомпиляции. 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 прекомпиляция заголовков не нужна. А вот в приложении по Windows – уже гораздо более существена.
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.
3. Пока эта информация не нужна, но на всякий случай, из расчета на следующий семестр. Если пишется приложение под Windows с использованием библиотеки MFC, то существенно, как подключать эту библиотеку. По умолчанию (это не относится к консольному приложению) в свойствах стоит Configuration Properties|General|Use of MFC –in a shared DLL. Плюсы такого выбора (когда DLL не включается в exe-файл) – маленький размер exe-файла. Минус – на компьютере без установленной DLL программа не запустится. Поэтому, если вы отдаете программу (exe-файл) кому-то, скомпилируйте его в режиме Use of MFC – in a static library (предварительно поменяв конфигурацию на Release).

Редактор

Удобные приемы:

1. Выделение строки с помощью клика мышью слева и выделение слова двойным кликом.
2. Вставить/убрать комментарии можно с помощью кнопочек с нарисованными строчками (или Edit|Advanced – Comment (un)selection. Выделять лучше строки целиком, проведя мышкой с нажатой левой кнопкой по левой границе текста. Тогда комментарий вставится построчный.
3. Текст должен быть отформатирован. Руками вставлять пробелы не надо. Отступы или делаются автоматически, или выделенный текст можно отформатировать с помощью Edit|Advanced – Format Selection (или соответствующее сочетание клавиш).
4. Удобная возможность – перейти к объявлению (прототипу) или к определению функции (реализации). Щелкаем правой кнопкой над функцией 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.
5. <Ctrl + Space> - автодополнение. Дополняет идентификаторы и поля объектов. Умное, но не дополняет ключевые слова.

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 - ...

Отладка

Отлаживать программу необходимо цивилизованным путем. А именно, с помощью Debugger: точек остановки, называемых Breakpoints (клик мышкой по серой полоске слева от строчки) и перехода по F5 (или кнопке с треугольничком) от одной точки остановки до другой. При этом значения переменных можно посмотреть, подведя мышь к нужной переменной или внизу в спец. окошке (locals). Есть окошко Breakpoints, с помощью которого можно ими управлять (если нет, то Debug-Window-Breakpoints). Также бывает полезна пошаговая отладка. Для нее есть панель Debug (если нет, то View – Toolbars – Debug) и там нужно использовать Step over (переход на след.строчку без захода внутрь функции), Step in (с заходом), Step out (выход из функции, в которую зашли). Пошаговую отладку можно чередовать с передвижением по breakpoints.

В дальнейшем добавятся еще два способа отладки: при прерывании программы в процессе выполнения в режиме debug полезно пользоваться окном Call Stack, где можно просмотреть последовательность вызовов функций, приведшую к прерыванию (ошибке). Также, при создании графического приложения для отладки используется макрос TRACE, с помощью которого можно делать вывод в окошко Output.

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.
На данный момент советую держать все файлы в одной директории, иначе надо отслеживать, чтобы в include был прописан правильный относительный путь и при подготовке архива с исходниками структура директорий сохранялась.

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). Естественно, что для build все равно, какой файл в данный момент высвечен в окне.

(!) 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, *.vcxproj, *.sln (если программа под Windows, то еще и *.rc и директория /res/ с ресурсами).