



ПРОГРАММИРОВАНИЕ  
**НА ЯЗЫКЕ C**

# Урок №1

Программирование  
на языке

C

## Содержание

1. Предварительные рассуждения .....	3
2. Инсталляция Microsoft Visual Studio 2013 .....	9
3. Первый проект .....	18
4. Вывод данных .....	31
5. Типы данных .....	37
6. Переменные и константы .....	41
7. Ввод данных .....	50
8. Литералы .....	54
9. Домашнее задание .....	56

# 1. Предварительные рассуждения

## **Вступительное слово.**

Добро пожаловать, в мир программирования! Что такое программирование? Наверное, каждый из Вас когда-то слышал это слово. Перефразировав известную цитату, можно сказать так: «Программист — это звучит гордо!» И это действительно так. Если Вы, бороздя просторы Internet, наталкивались на сайты с предложением о работе, то обращали внимание на размер заработной платы, которая предлагается программистам. Естественно, тут же возникает вопрос: почему такой высокий уровень оплаты труда программиста? Все согласно законам рынка: есть спрос на программистов, но, на данный момент, количество хороших специалистов слишком мало. Конечно, если бы программистом можно было стать за короткий срок, заучив ряд слов, профессия вряд ли была такой популярной. Но не расстраивайтесь! Мы, Ваши преподаватели, постараемся сделать все от нас зависящее для того, чтобы не только рассказать Вам как программировать, а научить Вас жить программируя. Однако, наша с Вами совместная работа должна быть взаимной. Преподаватель, как бы он не старался, не сможет обучить студента, если последний этого не захочет. Работа, только постоянная работа над собой сможет провести Вас к вершине мастерства.

Искренне надеемся на то, что Вы увлечетесь волшебным миром программирования!!!

Прежде чем приступать к изучению любой науки, всегда необходимо выяснить, откуда эта наука появилась, как она развивалась, каковы её корни и значение в истории.

## **Исторические факты**

Английский математик XIX века Шенкс потратил более 20 лет своей жизни на вычисление числа  $\Pi$  с точностью 707 значащих цифр после запятой. Этот результат получил славу рекорда вычислений XIX века. Однако впоследствии было обнаружено, что Шенкс ошибся в 520-м знаке, и поэтому все последующие значащие цифры были вычислены неверно.

В 1804 году французский изобретатель Жозеф Мари Жаккар создал «программно-управляемый» ткацкий станок. Для управления станком использовались перфокарты, соединенные друг с другом в виде ленты. Деревянные шпильки «читающего устройства» станка по расположению отверстий в перфокарте определяли, какие нити следует поднять, а какие опустить для получения нужного узора.

В 1890 году в США изобретателем Германом Холлеритом разработана электромеханическая счетная машина - табулятор, управляемая перфокартами. Она была использована для составления таблиц с результатами переписи населения США. Основанная Холлеритом фирма по производству табуляторов впоследствии превратилась в корпорацию International Business Machines (IBM).

В 1936 году двадцатипятилетний студент Кембриджского университета англичанин Алан Тьюринг опубликовал статью «О вычисляемых числах», в которой рассматривалось гипотетическое устройство («машина

Тьюринга»), пригодное для решения любой разрешимой математической или логической задачи, — прообраз программируемого компьютера.

В 1941 году немецкий инженер Конрад Цузе построил действующий компьютер Z3, в котором использовалась двоичная система счисления. Программы записывались на перфоленте.

В 1945 году в высшем техническом училище Пенсильванского университета (США) физик Джон Мочли и инженер Проспер Экерт построили полностью электронную машину «Эниак». Для задания программы было необходимо вручную установить тысячи переключателей и воткнуть сотни штекеров в гнезда контактной панели.

1 июня 1945 года был разослан отчет американского математика венгерского происхождения Джона фон Неймана «Предварительный отчет о машине Эдвак», содержащий концепцию хранения команд компьютера в его собственной внутренней памяти.

21 июня 1948 года в Манчестерском университете (Великобритания) на машине «Марк-1» выполнена первая в мире хранимая в памяти машины программа — поиск наибольшего сомножителя заданного числа.

В 1949 году под руководством Мориса Уилкса создан компьютер «Эдсак». Проектировщики «Эдсака» ввели систему мнемонических обозначений, где каждая машинная команда представлялась одной заглавной буквой, и автоматизировали настройку подпрограмм на определенное место в памяти. Морис Уилкс назвал мнемоническую схему и библиотеку подпрограмм собирающей системой (assembly system) — отсюда слово «ассемблер».

В 1949 году в Филадельфии (США) под руководством Джона Мочли создан «Краткий код» — первый примитивный интерпретатор языка программирования.

В 1951 году в фирме Remington Rand американская программистка Грейс Хоппер разработала первую транслирующую программу. Хоппер назвала ее компилятором (compiler — компоновщик).

В 1957 году на 20-м этаже штаб-квартиры фирмы IBM на Мэдисон-авеню в Нью-Йорке родился язык Фортран (FORMula TRANslation — трансляция формул). Группой разработчиков руководил 30-летний математик Джон Бэкус. Фортран — первый из «настоящих» языков высокого уровня.

В 1963 году был создан язык программирования Бейсик. Основателями языка стали Джон Кемени и Томас Курт, сотрудники Дартмут Колледжа. Под руководством создателей язык был реализован группой студентов колледжа. Самый первый диалект языка назывался Dartmouth BASIC.

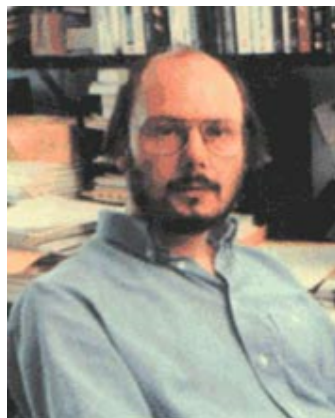
С 1958–1968 годы велись разработка и усовершенствование языка программирования под названием Алгол, название которого произошло от словосочетания «алгоритмический язык» (algorithmic language). В отличие от Фортрана, в основном используемого в США и Канаде, Алгол был широко распространен в Европе и СССР. Язык был создан международным комитетом, в составе которого были европейские и американские ученые — Джон Бэкус, Питер Наур, Уолли Фойрцойг, Никлаус Вирт.



В 1970 Никлаус Вирт, создал язык программирования, название которому дал в честь французского физика и математика Блеза Паскаля. Паскаль планировался Виртом, как язык, обучающий процедурному программированию.

В 1972 году 31-летний специалист по системному программированию из фирмы Bell Labs Деннис Ритчи разработал язык программирования Си.

Первое описание языка было дано в книге Б. Кернигана и Д. Ритчи, которая была переведена на русский язык. Долгое время это описание являлось стандартом, однако ряд моментов допускали неоднозначное толкование, которое породило множество трактовок языка С. Для исправления этой ситуации при Американском национальном институте стандартов



(ANSI) был образован комитет по стандартизации языка С

В 1983 году был утвержден стандарт языка С, получивший название ANSI C.

В начале 80-х годов в той же Bell Laboratory Бьерном Страуструпом в результате дополнения и расширения языка С был создан новый по сути язык, получивший название «С с классами». В 1983 году это название было заменено на С++.

23 мая 1995 года компания Sun Microsystems выпустила новый язык программирования под названием Oak. Язык был разработан для программирования бытовой электроники. В последствии Oak был переименован в

язык Java и начал широко использоваться в разработке приложений и серверного программного обеспечения.

В 2000-2001 годах был принят и стандартизирован новый язык программирования C# (си-шарп), специально разработанный для платформы .NET. В создании языка принимали участие сотрудники Microsoft Research (НИИ при корпорации Microsoft) — Андерс Хейлсберг, Скотт Вилтамут, Питер Гольде и другие известные специалисты, в том числе Эрик Майер. Версия языка — C# 2.0 была предъявлена весной 2005 года. Следует отметить, что одним из языков, на базе которых основан C# , был всё тот же старый добрый C++.

Итак, из краткого перечня исторических фактов мы выяснили, что C и C++ два различных языка программирования, даже не смотря на то C++ основан на базе C. Наши занятия будут посвящены изучению обоих этих языков программирования. Естественно, получение знаний будет последовательным и сначала мы познакомимся с языком C и, затем плавно перейдем к C++.

После того, как мы совершили небольшое путешествие в прошлое, мы можем переходить к настоящему. Для дальнейшей работы нам с вами безусловно понадобится программное обеспечение. В связи с быстрым развитием рынка IT-технологий постоянно создаются более новые версии различных программ. В последующих разделах урока мы рассмотрим установку программного продукта, который мы будем использовать в рамках процесса обучения.



## 2. Установка Microsoft Visual Studio 2013

Microsoft Visual Studio — набор программных продуктов от компании Microsoft, которые применяются для разработки решений различного масштаба: от лабораторных работ студентов до проектов корпоративного уровня. В состав Visual Studio входит интегрированная среда разработки (IDE — Integrated development environment), многочисленные инструментальные средства и утилиты. С помощью Visual Studio можно создавать как консольные приложения, так и приложения со сложным графическим интерфейсом. Мы будем использовать продукты из линейки Visual Studio в рамках нашего процесса обучения.

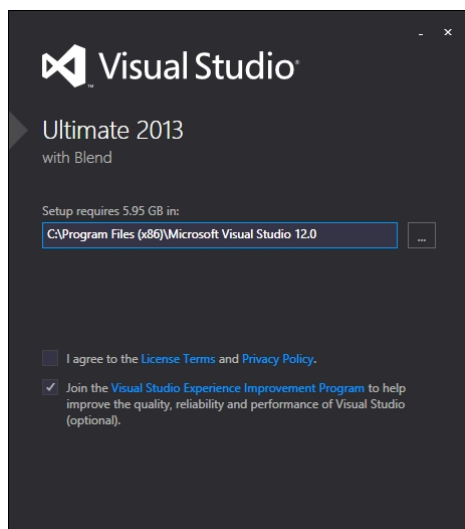
История Visual Studio началась в 1997 году. Именно тогда была выпущена Visual Studio 97. Вы будете изучать программирование с использованием Microsoft Visual Studio 2013. Это самая актуальная версия на текущий момент. Перед тем как рассмотреть установку Visual Studio разберем вопрос редакции (edition) Visual Studio. Редакция Visual Studio — это версия Visual Studio, которая отличается от другой редакции набором возможностей. У Visual Studio 2013 есть следующие редакции: Ultimate, Premium, Professional, Test Professional, Express Editions.

Ultimate — это самая полная редакция Visual Studio. У каждой редакции своя ценовая политика. Исключением является Express Editions. Это набор Visual Studio (й), заточенных под конкретные нужды. Например, для разработ-

ки мобильных решений или веб-решений, или решений рабочего стола и т.д. Использовать Express Editions можно абсолютно бесплатно, в том числе и для коммерческой разработки. Важно отметить, что различные редакции Visual Studio доступны на разных языках. Например, у вас может возникнуть желание установить русскоязычную локализацию. Не делайте этого! В рамках вашей будущей профессиональной деятельности вы будете сталкиваться с коллегами и проектами из других стран, которые наверняка будут использовать англоязычный интерфейс. Именно поэтому всегда предпочитайте язык оригинальной версии — английский.

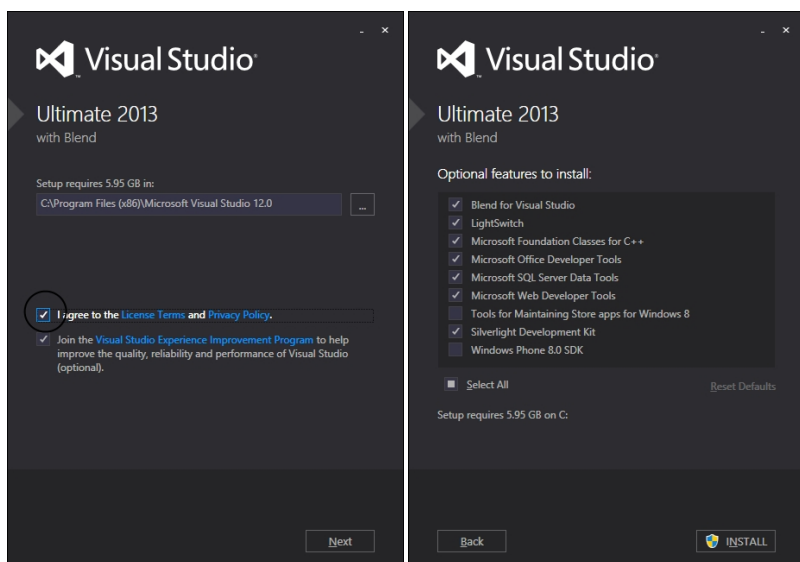
Мы рассмотрим установку Visual Studio на примере Microsoft Visual Studio 2013 Ultimate и Microsoft Visual Studio Express 2013 для Windows Desktop. Начнем с Ultimate.

Для того, чтобы запустить инсталляцию Ultimate, вставьте DVD диск в DVD-привод (или смонтируйте ISO-образ DVD диска с помощью, например, программы Daemon Tools Lite) и запустите файл vs\_ultimate.exe, после чего перед вами появится следующее окно: В этом окне вы можете выбрать путь для инсталляции Visual Studio, определиться хотите ли вы

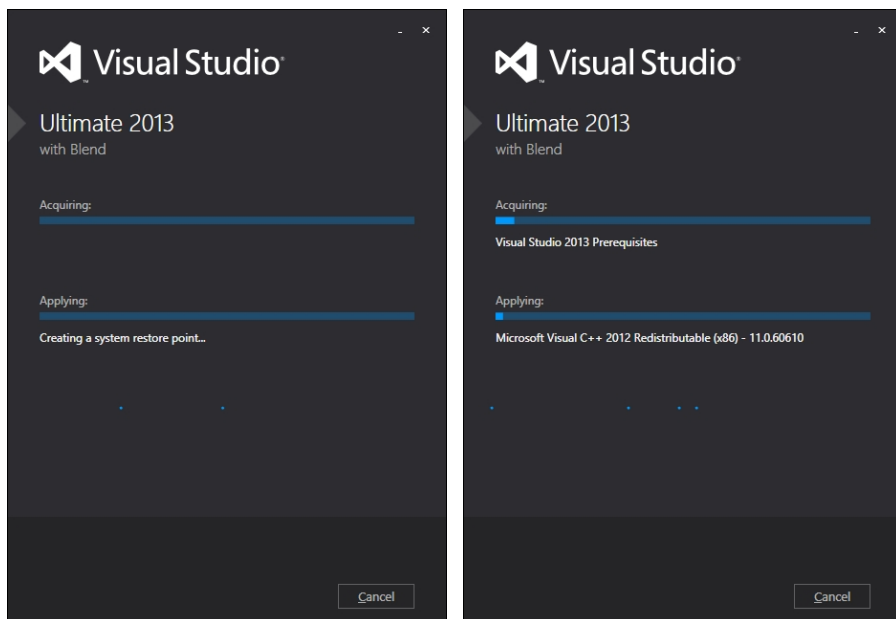


участвовать в программе улучшения Visual Studio. Если вы хотите участвовать в программе улучшения качества, то оставьте галочку в опции «Join the Visual Studio Experience Improvement Program ...». Для того, чтобы начать установку Visual Studio нужно будет выбрать пункт «I agree to the License Terms and Privacy Policy».

Выбрав эту опцию вы подтверждаете тем самым, что ознакомились и согласились со всеми пунктами данной лицензии. Если вы сделали правильный выбор внизу окна у вас появилась кнопка «Next». Нажмите на неё. После чего вы увидите окно, которое позволяет вам выбрать дополнительные программные продукты для установки. На этом этапе можете оставить выбранные по умолчанию продукты. В дальнейшем у вас будет возможность доустановить компоненты. Для продолжения процесса нажмите на кнопку «INSTALL».

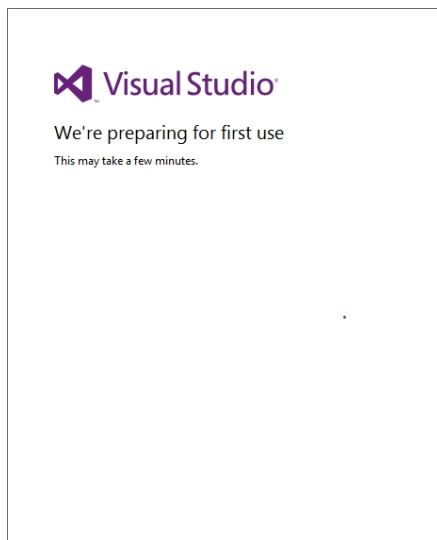
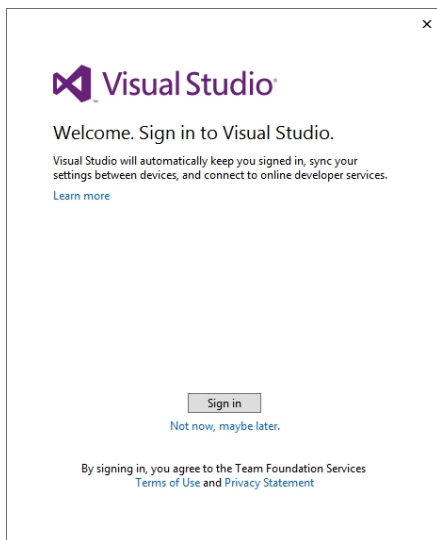
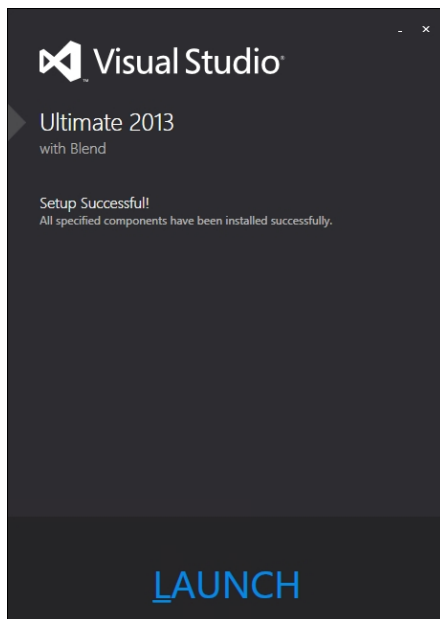


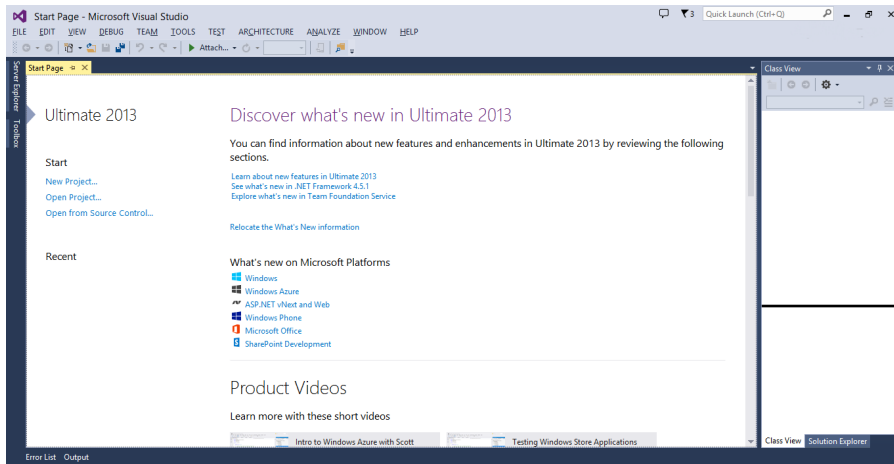
У вас должен стартовать процесс инсталляции. Он займет некоторое время, в течении, которого вы сможете наблюдать за прогрессом в работе. Ниже мы приведем несколько характерных для этого состояния окон.



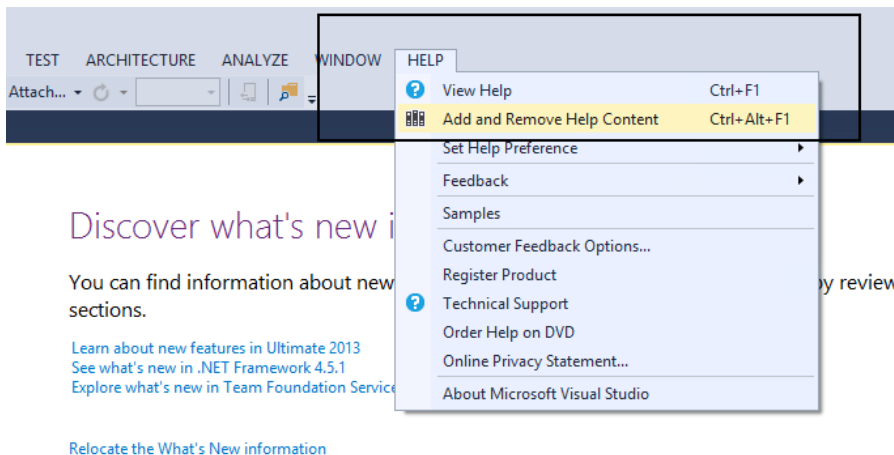
В конце установки если всё прошло успешно вы увидите окно с кнопкой «LAUNCH». Нажав на неё вы сможете запустить Visual Studio в первый раз, также вы можете использовать привычный интерфейс запуска программ Windows (меню Пуск, ярлыки и т. д.)

Запустим Visual Studio, чтобы убедиться в успешности наших предыдущих действий. Во время первого запуска вас могут попросить ввести ваш Microsoft Live ID. При этом у вас есть возможность не вводить его. Для этого необходимо выбрать пункт «Not now, may be later».

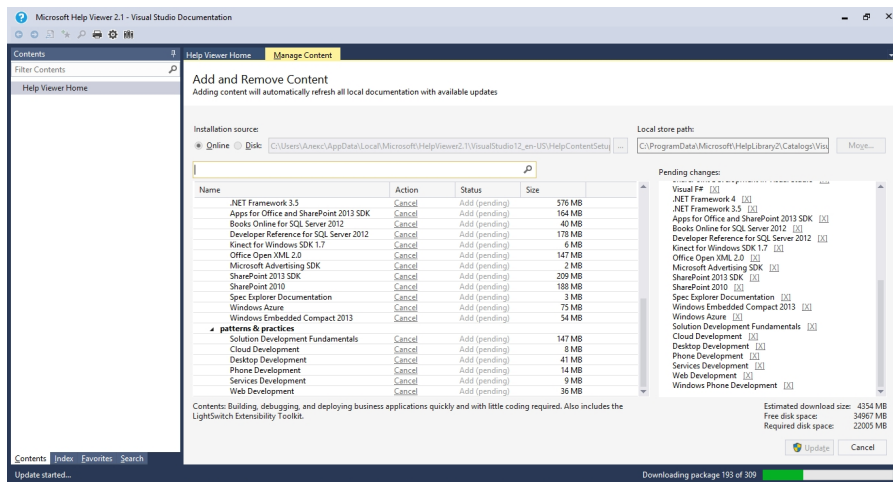




При обучении и разработке приложений вам очень пригодится справочная информация (MSDN) от Microsoft. Для того, чтобы установить локальную копию справки нужно выбрать пункт меню **Help->Add and Remove Help Content** или же нажать **Ctrl+Alt+F1**.

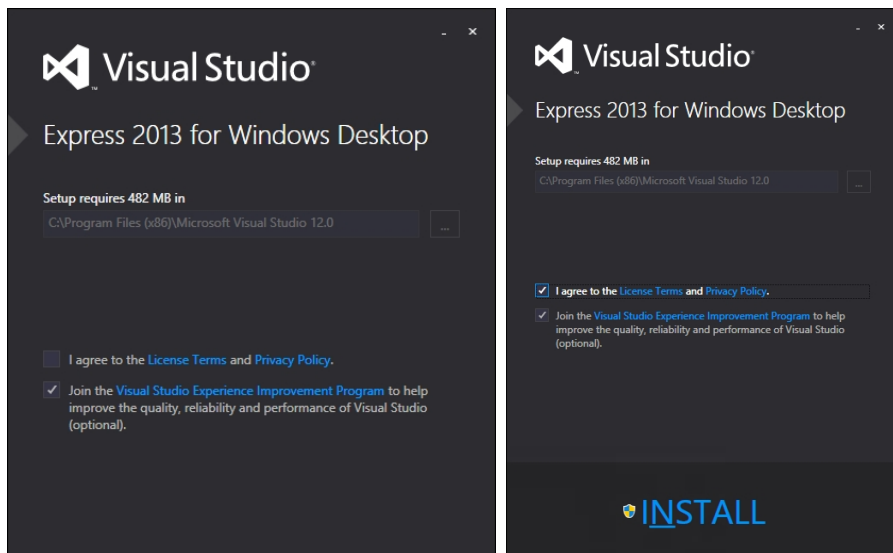


Вы увидите окно с выбором справочного материала. Рекомендуем везде нажать ссылку «Add». Справки много не бывает :) После выбора материалов нажмите кнопку «Update» для их локальной установки.



Теперь рассмотрим процесс инсталляции Microsoft Visual Studio Express 2013 для Windows Desktop. Он очень схож с инсталляцией Ultimate.

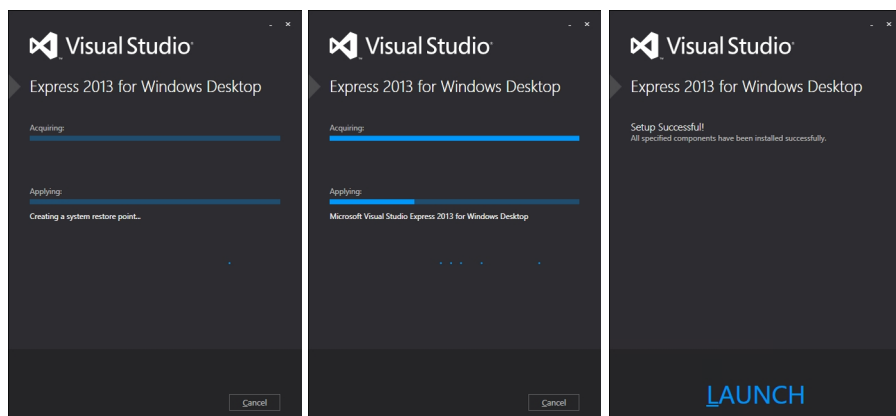
Для того, чтобы запустить инсталляцию Express, вставьте DVD диск в DVD-привод (или смонтируйте ISO-образ DVD диска с помощью, например, программы Daemon Tools Lite) и запустите файл **wdexpress\_full.exe**, после чего перед вами появится следующее окно:



И опять же в этом окне вы можете выбрать путь для инсталляции Visual Studio, определиться хотите ли вы участвовать в программе улучшения Visual Studio. Если вы хотите участвовать в программе улучшения качества процедуры инсталляции данного продукта, то оставьте галочку в опции «Join the Visual Studio Experience Improvement Program». Для того, чтобы начать установку Visual Studio нужно будет выбрать пункт «I agree to the License Terms and Privacy Policy».

Выбрав эту опцию вы подтверждаете тем самым, что ознакомились и согласились со всеми пунктами данной лицензии. Если вы сделали правильный выбор внизу окна у вас появилась кнопка «INSTALL». Нажмите на неё и у вас начнется процесс установки Visual Studio. У вас должен стартовать процесс инсталляции. Он займет некоторое время, в течении, которого вы сможете наблюдать за прогрессом в работе. Ниже мы приведем несколько характерных для этого состояния окон.





В конце установки если всё прошло успешно вы увидите окно с кнопкой «LAUNCH». Нажав на неё вы сможете запустить Visual Studio в первый раз, также вы можете использовать привычный интерфейс запуска программ Windows (меню Пуск, ярлыки и т. д.). Как и при установке Ultimate редакции вас попросят ввести свой LIVE ID (его можно не вводить), также не забудьте установить справочную систему.

Если вы прошли все пункты установки успешно, то вы уже готовы к тому, чтобы написать свою первую программу.

Вот и все. Теперь вы можете переходить к следующим разделам урока, где мы будем учиться программировать.

### 3. Первый проект

Один человек по имени Сергей, описывая свои впечатления о Праге написал мне:

«Если вы попали на официанта не говорящего по-русски, то берите меню и показывайте пальцем что вы хотите. Иногда похожие слова обозначают разные вещи. Например, слово фрукты по-чешски звучит почти как овощи. Угадайте что вам принесут, если вы попросите овощной салат?:)»

Неизвестный Вам Сергей хотел предостеречь своих друзей о возможной ошибке, но вряд ли он смог бы помочь чем-нибудь, если бы вместо официанта, нам приходилось бы общаться с компьютером. Последнему, увы, неведомо, в какую строку меню, Вы тычете пальцем. Компьютер педантично следует лишь четким указаниям. И указания эти должны даваться с помощью специальных команд. Команды в свою очередь составляют целые самостоятельные языки. Языки, понятные компьютеру — языки программирования. Вывод: чтобы найти общий язык с официантом-чехом надо быть человеком находчивым. Чтобы найти общий язык с компьютером, надо этот язык знать.

Из первого раздела Вам уже известно, что C — это язык программирования. Язык, который дает нам возможность вразумительно объяснить компьютеру, что мы от него хотим. Хотя, если быть до конца откровенными, компьютер понимает только один язык — язык машинных кодов. Для примера, программа, которая выводит на экран фразу «Hello, world!», выглядит приблизительно так на «родном» для компьютера языке:

```

_YH!Ÿ+ .5lЧN+f-H!Ÿ+ +f+ Ɂд-Ɂ+fO_fO!Ɂ+ɁNɁ+Ɂ+!ФN+Ɂ-;-wGГ=xФN+MЗР __ + @ A -! | -!+ L-!T
u . .5lЧN+f-&JZŸQW&бФбN+&JL &JLf&OB_ФN
+f;sJl1ФN+JLl1t; _t=ИФN+ GŸu_OŸWfM+-чfJN3+M-Ac°
>] J!;J+JlMJlм _сбшбN+Jl 9шФN+t$Jl$рФN+шN) fd! __+RРбшб
N+Jl 9шФN+тJl$рФN+шC) + +э
тAc°-ш3-Jl_ФN+f;s $lФN+Jl +u"бAФN+Ɂ+MAc°ŸD At 3+- 3+
ыŸы•ы°ыŸыJlD$П$. _4ЧN++ . _дJl4 Ÿ- ubSJl др4 +__e --fУЙf-+
-+ЙA+@УN+Jl-[Xf +tRPRVh•Ÿ+ш_S Z+efO-fJлцf-дɁ+$ RfRfh ш4
Jl+Zы!SшCɁд-ы!Ɂ+ьM-Ɂ+Ɂ+e fM+fO_fO!fɁбJлЁdŸ т3+OшAc°Jl$AФN+-
д64 ЙдЙ$4 f_f_t дJl4 Ÿ- uSSJl др4 +__e --fУЙf-++ЙA+@УN+Jl-[
+eЙ]NfM+fO-fJлц+ыf-дɁ+$ RfRfh шз Jl+Zы+Sш!
$ PE L |7 p Ɂ!
` P oK P Ÿ+ 0

```

Вы скажите что в таком коде писать невозможно и будете абсолютно правы!! Но, так мы писать и не будем. Для этого и нужен язык программирования, чтобы облегчить составление программ. Языки программирования делятся на две основные группы ИНТЕРПРЕТИРУЕМЫЕ и КОМПИЛИРУЕМЫЕ. Такое деление связано с тем, какая специализированная программа переводит команды с языка программирования на машинный язык — КОМПИЛЯТОР или ИНТЕРПРЕТАТОР. Давайте выясним в чем же разница между ними. Представим, что у нас есть файл, в котором содержится набор команд

### Ситуация первая. Команды написаны на интерпретируемом языке.

Каждый раз при запуске программы интерпретатор осуществляет проверку кода построчно. Если ошибок в синтаксисе нет, команды будут преобразованы в машинный код (набор инструкций для процессора). Программа запустится на выполнение. Если есть ошибка, интерпретатор остановится и Вам будет предложено ее исправить и запустить программу снова. Но, даже если ошибок больше нет и программа

окончательно дописана Вами, при каждом её запуске будет срабатывать интерпретатор и опять осуществлять проверку кода. Таким образом, можно сделать выводы, что машинная версия кода нигде не сохраняется. Минусы такого подхода состоят в том, что скорость запуска программы снижается, но отключить проверку невозможно.

### **Ситуация вторая. Команды написаны на компилируемом языке.**

Компилятор действует почти также, как и интерпретатор, т. е. проверяет код построчно. Но если встречается ошибку, то не останавливается, а исследует код до конца, выявляя все последующие ошибки и выдавая о них сообщения. Кроме того компилятор формирует специальный объектный файл с расширением .OBJ. В этом файле хранится текст программы переведенный на машинный язык.

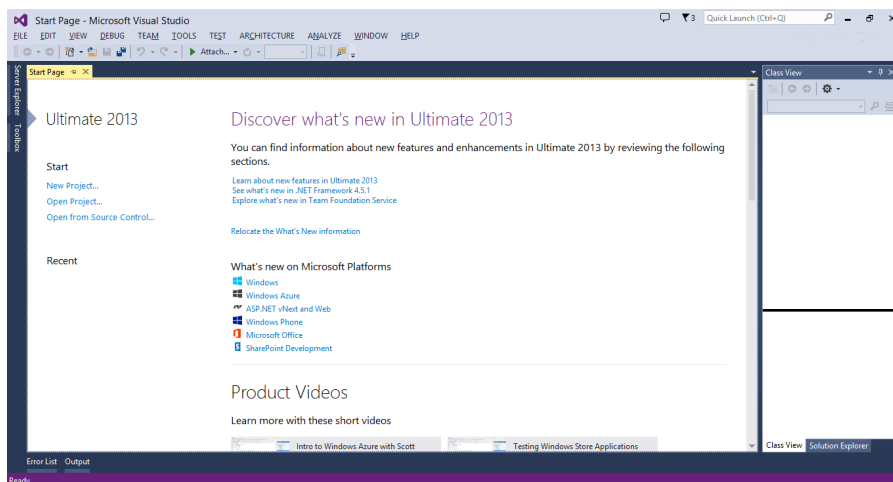
Однако, компьютер не работает напрямую с этим файлом. Существует такое понятие, как компоновка или линковка. Линковщик — это еще одна специальная программа, которая собирает машинный код (из файла с расширением .OBJ) и различные вспомогательные данные в единый исполняемый файл с расширением .EXE. Такой файл может быть запущен на выполнение как отдельная, самостоятельная программа и в его запуске компилятор уже не принимает никакого участия.

Язык С, к изучению которого вы приступаете является компилируемым языком. В нашем случае, при работе с оболочкой Microsoft Visual Studio вызов компилятора осуществляется автоматически и позволяет переводить команды языка С в машинный код, что называется — «легким движением руки».

## Проба пера

Один из основателей языка С, Брайан Керниган, сказал: «Единственный способ изучать новый язык программирования — писать на нем программы». Чем мы с Вами и начнем, сейчас, заниматься. Так уж повелось в мире программирования, что первой программой на новом языке является программа «Hello world!».

Для того, чтобы написать свою первую программу, для начала необходимо запустить ярлык программы Microsoft Visual Studio 2013 из пункта меню «Пуск» -> «Все программы» -> «Microsoft Visual Studio 2013» или другим привычным для вас образом. После запуска мы увидим изображение, подобное представленному на изображении:



Сейчас, попробуем создать проект, который и будет, в конечном итоге, представлять нашу программу. Детальней, что такое проект, мы разберемся попозже, когда будем писать большие программы. Пока будем представлять

себе проект как объединение нескольких файлов. А теперь давайте по шагам:

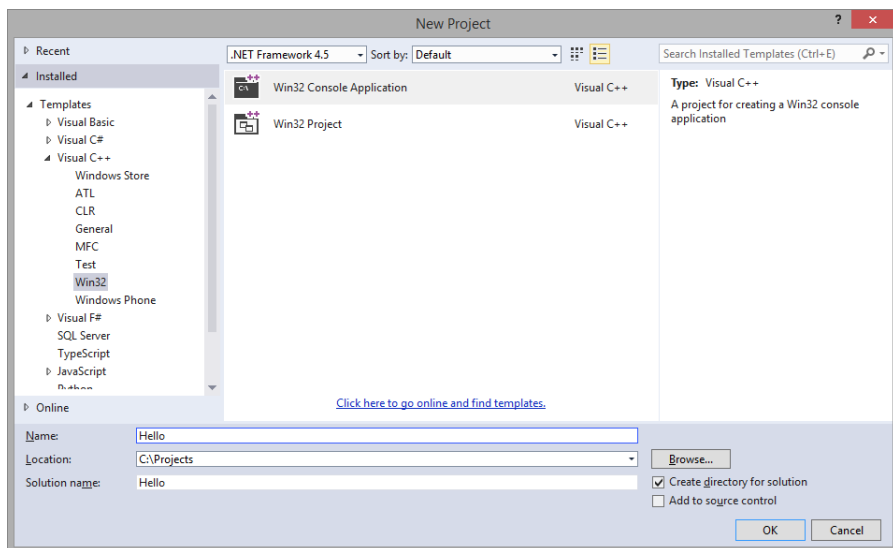
- После того, как Вы загрузили оболочку, выберите пункт меню File -> New -> Project. Перед Вами появится диалоговое окно.

- В открывшемся диалоговом окне New Project (новый проект) в списке Installed Templates (шаблоны) выберите тип проекта Visual C++ -> Win32 -> Win32 Console Application

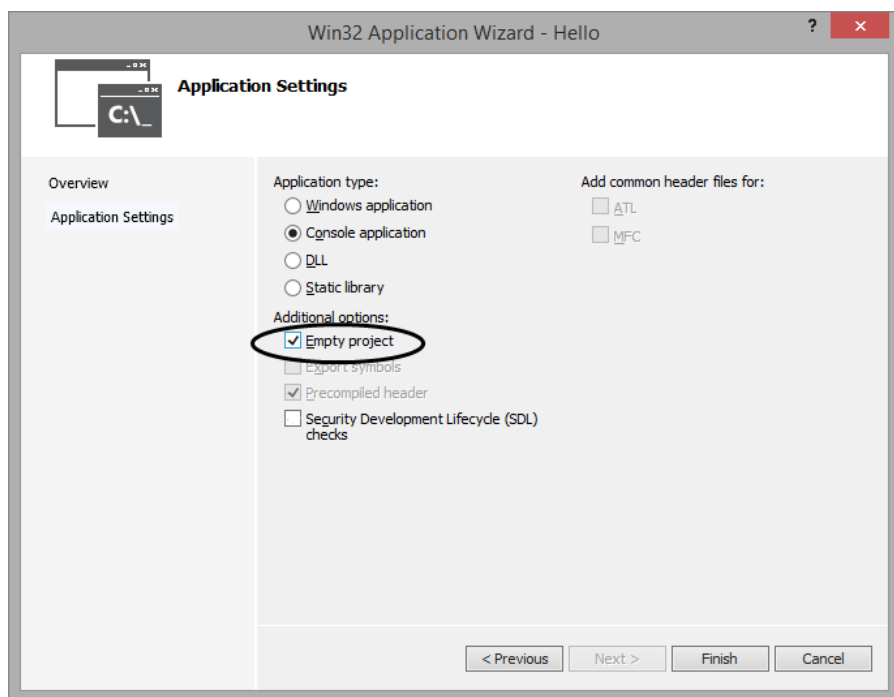
- В поле Location укажем, на каком диске и в какой папке будет находиться наш проект. Для этого введите в этом поле C:\Projects (или удобное Вам имя папки, где будут храниться все Ваши домашние проекты)

- Дадим имя нашему ПРОЕКТУ — для этого введите в поле Name имя проекта Hello.

- Теперь можно нажать кнопку ОК.



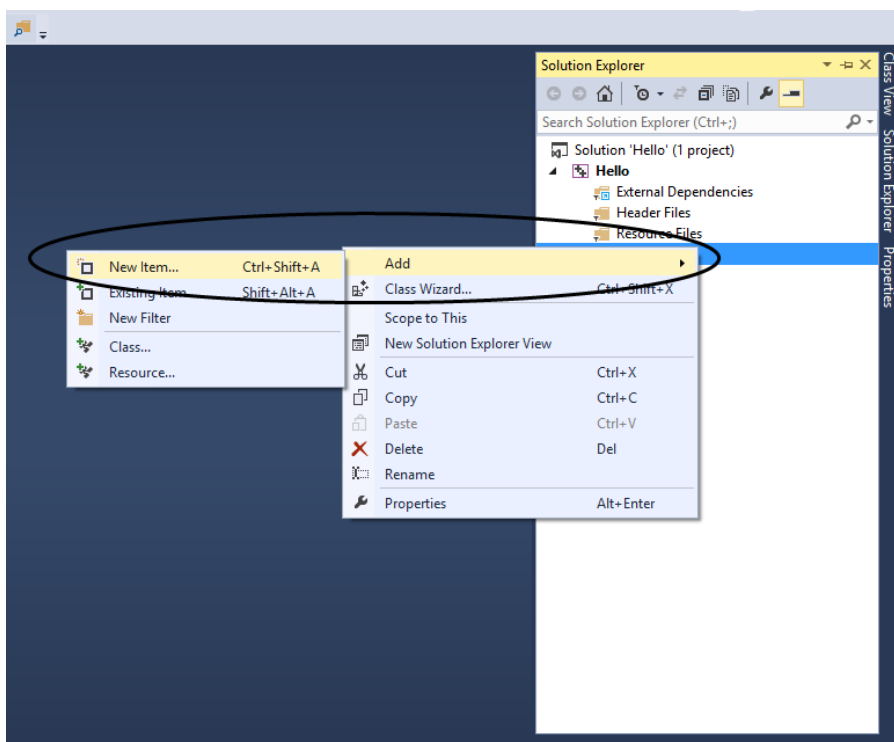
Перед Вами открылось окно настройки свойств проекта — выберите вкладку **Application Settings**. Установите галочку в поле **Empty Project** — это значит, что мы создаем пустой проект. Также снимите галочку **Security Development Lifecycle (SDL) checks**. Теперь нажимайте кнопку **Finish**.



Итак, мы подготовили место для размещения нашей программы. Не останавливаясь на достигнутом, добавим в проект чистый файл. В нем будем набирать текст нашей программы. Для этого необходимо выполнить следующие действия:

■ Справа появилось окошко под названием Solution Explorer. Для вызова этого окна можно также использовать клавиатурную комбинацию Ctrl+Alt+L. В этом окне Вам необходимо щелкнуть правой кнопкой на папке под названием Source Files.

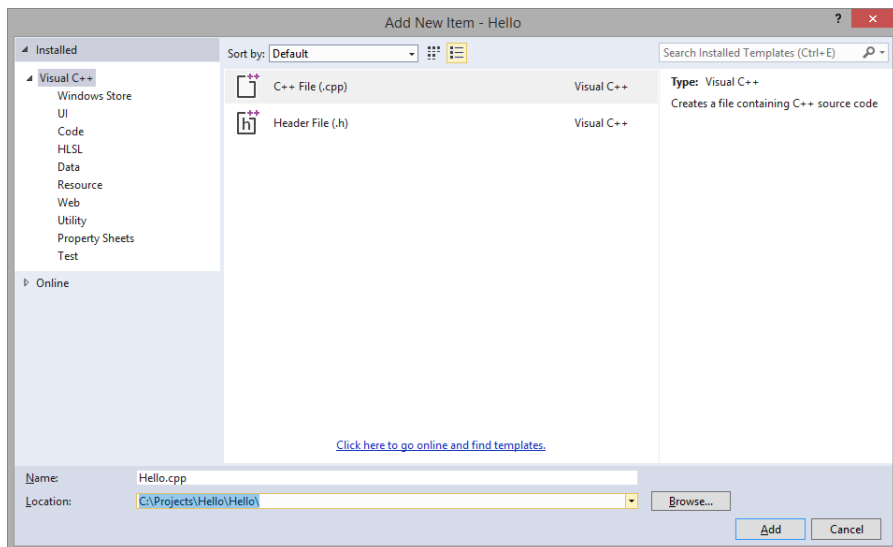
- В выпавшем меню выбираем Add->Add New Item...



■ Открывается окно выбора файлов. Опять у вас огромный выбор. Рекомендуем сейчас выбрать значок C++File (.cpp) (файл, содержащий программу на языке C/C++).

■ В текстовом поле Name (имя файла) введите имя файла Hello. Нажмите кнопку Add.





После нажатия на кнопку Add у вас появится текстовая область, где вы сможете набрать свою первую программу.

## Пример первой программы на языке C.

Перед тем как начать писать программу, давайте введем для удобства понятие комментария. Комментарий — это заметки к программе, которые предназначены исключительно для программиста. Компилятор игнорирует их. Например, с помощью комментария можно обозначить для чего используется та или иная строка программы. В нижеописанном примере указано, как правильно работать с комментариями.

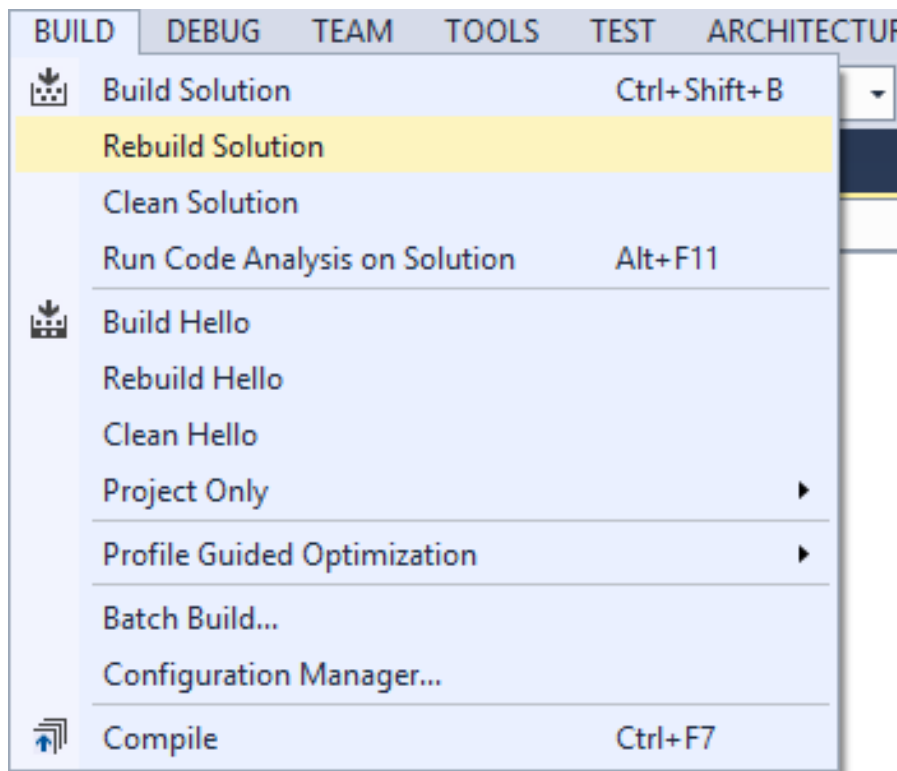
А теперь, в появившемся текстовом окне наберем следующий код:

```

// Это комментарии к программе
// Они выделяются зеленым цветом
// Начинаются комментарии двумя черточками //
/* если необходимо создать многострочный
комментарий, используется конструкция
*/ комментарий */
/* Данная строка подключает в программу библиотеку под названием
iostream. Библиотека – файл, в котором содержатся описания
различных функций, реализованных другими программистами.
Данная программа получила возможность использовать функции
находящиеся в библиотеке iostream */
#include <iostream>
/* В языке C++ существует понятие пространство имен.
Это пространство определяет некую область, на которую
приходятся действия оператора или функции. Для того, чтобы использовать
оператор, находящийся в определенном пространстве,
необходимо подключить это пространство в свою программу.
Ниже подключается пространство под названием std */
using namespace std;
void main() // Начало программы, отсюда программа начнет своё выполнение
// Весь текст программы располагается между фигурными скобками
{
    //Это фигурная скобка
    // Следующая строка выводит на экран приветствие Hello, World!
    // Данное действие осуществляется с помощью cout<< Именно для его
    // работы подключалась библиотека и пространство имен, в которых он располагается
    cout<<"Hello, World!\n";
    // В конце команды стоит точка с запятой. Этим знаком ДОЛЖНА заканчиваться
    // каждая команда в языке C.
}

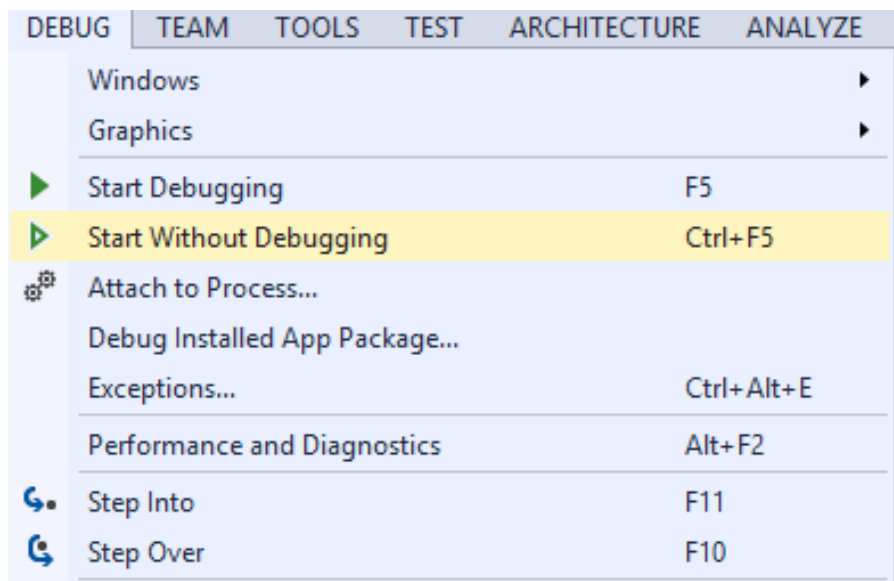
```

Как вы уже знаете, компьютер понимает только язык машинных кодов. И прежде чем программа будет выполняться компьютером, нужно ее перевести на язык машинных кодов. Вы помните, что это будем делать не мы, а КОМПИЛЯТОР. Проекты, написанные на C, включают в себя много файлов. Прокомпилируем сразу все файлы, включенные в проект. Для этого в строке меню выберите пункт меню Build (построить), затем Rebuild Solution (перестроить все)

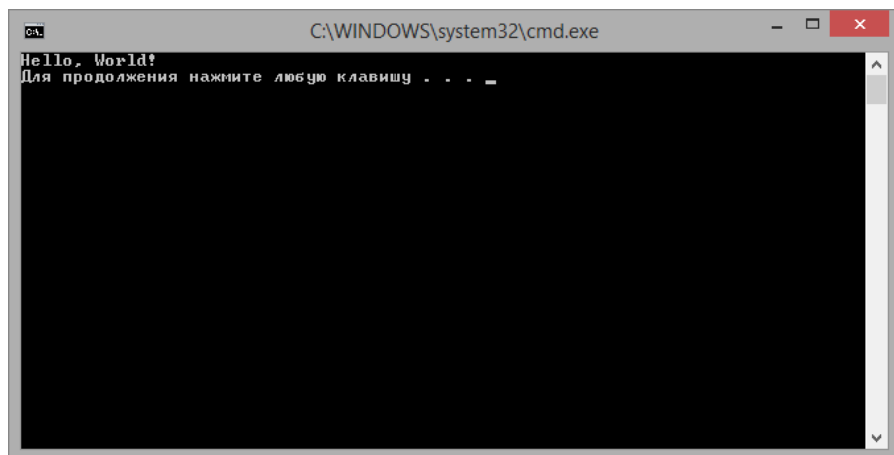


пределяет некую область, на которую

Надеемся, вы набрали текст без ошибок. Наша программа успешно переведена на язык машинных кодов и ее можно запустить на выполнение. Запустить программу на выполнение просто. В меню Debug выбрать Start Without Debugging.



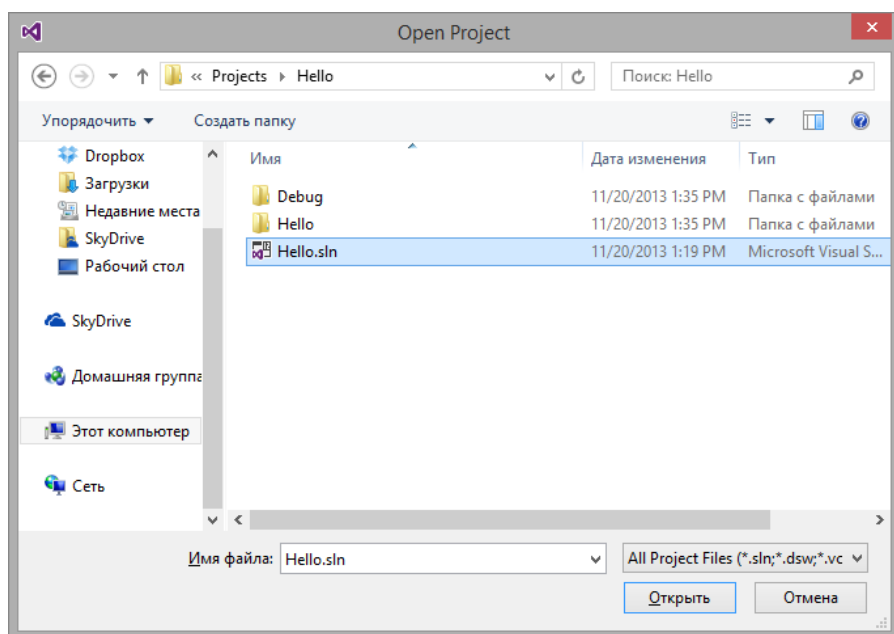
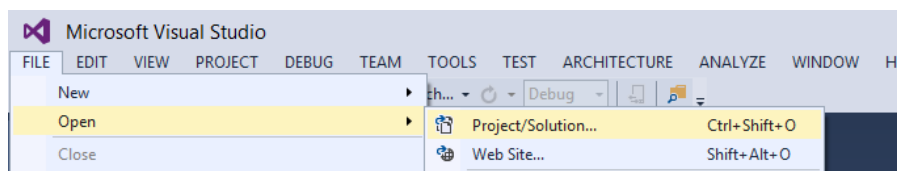
Программа отработает и вы увидите такое вот окно:



Поздравляю! Было довольно легко. Не правда ли? Для того, чтобы закончить работу программы нажмите любую клавишу на клавиатуре (Press any key to continue).

## Открытие сохраненного проекта

Для того чтобы восстановить ранее сохраненный проект на диске запустите Visual Studio (если она у Вас еще не загружена). Выберите в меню File пункт Open ->Project/Solution и укажите имя вашего проекта.



После того как проект открыт, Вы можете продолжить работу над ним. Проекты создаваемые в рамках урока вы сможете найти в подпапке урока с названием Sources. В частности этот проект находится по пути папка с уроком/Sources/Hello.

## 4. Вывод данных

Вы уже знаете, что с помощью команды `cout<<` мы можем выводить на экран различные текстовые строки.

Однако, чтобы компилятор понял такую команду, Вы должны помнить о следующих трех основных моментах:

1. В заголовке программы должна присутствовать строка `#include <iostream>`

2. Перед использованием команды необходимо подключить пространство имен, к которому принадлежит команда `cout`.

```
using namespace std;
```

3. Строку, которую мы хотим вывести на экран используя `cout<<`, мы обязательно записываем в кавычки. Например:

```
cout<<"здесь пишем то, что хотим";
```

Команда `cout<<` не только выводит на экран строки, но и позволяет их оформлять. Для оформления вывода строки используют специальные управляющие символы, представляющие собой комбинацию символа `\` и символа, определяющего действие, которое необходимо произвести над строкой. Эти управляющие символы называются Escape-последовательностями. Ниже приводятся некоторые из них:

<code>\b</code>	Удаление последнего выведенного символа
<code>\n</code>	Перейти на начало новой строки
<code>\t</code>	Перейти к следующей позиции табуляции
<code>\\</code>	Вывести обратную черту <code>\</code>
<code>\"</code>	Вывести двойную кавычку <code>"</code>
<code>\'</code>	Вывести одинарную кавычку <code>'</code>

Существование последних трех Escape-последовательностей сначала всегда вызывает легкое недоумение. Зачем пользоваться управляющими символами, если можно просто написать: `"` или `\` или `'`? Ответ лежит на поверхности, все эти три символа являются операторами и, если их «просто написать», то компилятор и будет воспринимать их как операторы. Например, когда слово используется в переносном смысле, оно заключается в кавычки. Предположим, Вам необходимо вывести следующий текст на экран:

```
The Man in red was "old friend" of John...
```

Если Вы не используете Escape-последовательности, то очевидно, что Ваша команда будет выглядеть так:

```
cout<<"The Man in red was "old friend" of John...";
```

И это приведет к неминуемой ошибке. Компилятор воспримет только часть строки, а именно `cout<<"The Man in red was"`. Двойную кавычку после `was` он посчитает закрывающей, а всё остальное примет, как неверный синтаксис языка. Такая программа, естественно, не запустится на выполнение. Правильный вариант таков:



```
cout<<"The Man in red was \"old friend\" of John...";
```

Теперь, давайте поговорим о том, где именно в `cout<<` можно указывать Escape-последовательности. Самое главное, что Вам необходимо знать, это — что Escape-последовательность всегда должна находиться внутри кавычек, т. к. является текстом, а дальше Ваши возможности практически не ограничены. Например, так:

```
cout<<" My name is"<<" - Ira\n ";
cout<<"I'm from Odessa\n ";
cout<<"My eyes are blue"<<"\n "<<"That`s all!!!";
```

В результате работы этой команды мы увидим на экране:

```
My name is - Ira
I'm from Odessa
My eyes are blue
That`s all!!!
```

## Практический пример использования `cout<<`

Напишем программу, которая выводит на экран краткую справку об изученных нами Escape-последовательностях. Вот, что мы хотим увидеть на экране:

<code>\b</code>	Backspace
<code>\n</code>	New line
<code>\t</code>	Horizontal tab
<code>\\</code>	Backslash \
<code>\"</code>	Double quotation mark "
<code>\'</code>	Single quotation mark '

Запускаем среду **Visual Studio 2013**. Создаем новый проект под именем **EscapeSequences**. Набираем код, который располагается ниже.

```
// Заголовок
#include <iostream>
// определение пространства имен, в котором есть cout<<
using namespace std;
// Главная функция
void main()
{
/* Следующая команда через 4 табуляции выводит текст
Escape Sequences
и переводит вывод на следующую строку */
                                cout<<"\t\t\t\tEscape Sequences\n";
                                // Выводит пустую строку
                                cout<<"\n";

/* Через 2 табуляции выводит текст \b,
и еще через 1 табуляцию Backspace
Затем \n переводит вывод на следующую строку */
                                cout<<"\t\t\b"<<"\tBackspace\n";
                                // Выводит пустую строку
                                cout<<"\n";

/* Через 2 табуляции выводит текст \n,
и еще через 1 табуляцию New line
Затем \n переводит вывод на следующую строку */
                                cout<<"\t\t\n"<<"\tNew line\n";
                                // Выводит пустую строку
                                cout<<"\n";

/* Через 2 табуляции выводит текст \t,
и еще через 1 табуляцию Horizontal tab
Затем \n переводит вывод на следующую строку */
                                cout<<"\t\t\t"<<"\tHorizontal tab\n";
                                // Выводит пустую строку
                                cout<<"\n";

/* Через 2 табуляции выводит текст \\",
и еще через 1 табуляцию Backslash \
Затем \n переводит вывод на следующую строку */
                                cout<<"\t\t\\\\"<<"\tBackslash \\ \n";
                                // Выводит пустую строку
                                cout<<"\n";

/* Через 2 табуляции выводит текст \t,
и еще через 1 табуляцию Double quotation mark "
Затем \n переводит вывод на следующую строку */
                                cout<<"\t\t\"\"<<"\tDouble quotation mark \"\n";
                                // Выводит пустую строку
                                cout<<"\n";

/* Через 2 табуляции выводит текст \',
и еще через 1 табуляцию Single quotation mark '
                                cout<<"\n";
```

```

Затем \n переводит вывод на следующую строку */
                                cout<<"\t\t'"<<"\tSingle quotation mark \'\\n";
// Выводит пустую строку
                                cout<<"\\n";
}

```

Откомпилируем программу (**Build -> Rebuild Solution**). Если возникло много ошибок, то вспомните следующие правила:

Если в программе будут выводиться сообщения на экран, то в начало программы записывается строчка `#include <iostream>` и подключается пространство имен, к которому принадлежит команда `cout` (`using namespace std;`)

Каждая программа должна содержать функцию с именем `main ()`. Работа программы начинается с выполнения этой функции.

Команды функции `main()` находятся внутри фигурных скобок `{ }`

Все команды обязательно должны заканчиваться символом точка с запятой.

И запустим ее (**Debug -> Start Without Debugging**).  
**P. S.**

Вы должно быть обратили внимание на то, что мы употребляем лишь латинские символы при выводе на экран данных. Это связано с тем, что программу мы с Вами писали в ОС Windows, а выполнение ее осуществляется в MS DOS. Дело в том, что каждый символ имеет в любой операционной системе свой числовой код. И система идентифицирует его именно по этому коду. Символьные коды кириллицы в MS DOS и Windows не совпадают, поэтому программа с использованием кириллицы будет работать не корректно. Например, написали мы в Windows:

```
cout<<"Утро";
```

А на экран выводиться:

```
µЄЁю
```

Это легко объяснить тем, что в Windows, например, буква о — 238, а в DOS этому коду соответствует буква ю. Кода же латиницы совпадают в обеих ОС. В последствии мы с Вами научимся исправлять эту ситуацию.

В стандарте языка C++ 11 программистам предоставили новую возможность для вывода специальных символов на экран. Для этого необходимо использовать «raw» строки. Объявление строки как «raw» даёт указание компилятору трактовать её посимвольно. Для того чтобы строку объявить такой строкой нужно использовать следующий формат.

```
R" (текст_строки) "
```

R указывает на то, что это «raw» строка. Содержимое строки обязательно заключать в скобки. Приведем несколько примеров:

```
cout<<R"(hello\nworld)"; // на экране hello\nworld
cout<<R>("Test 'string'\t"); // на экране "Test 'string'\t"
cout<<R"((Such brackets))"; // на экране (Such brackets)
```

## 5. Типы данных

После того, как Вы прочли предыдущие разделы урока, Вам уже ничего не стоит написать программу, которая выводит что-либо на экран.

```
// Заголовок
#include <iostream>
// определение пространства имен, в котором есть cout<<
using namespace std;
// Главная функция
void main()
{
    // вывод фразы "Поздравляем с хорошим началом!!! :)"
    // фраза выводится через три табуляции,
    // затем добавляются две пустых строки
    cout<<"\t\t\tCongratulation with good beginning!!! :)\n\n";
}
```

Вот собственно и всё, что Вы умеете(пока).

Человек никогда не должен останавливаться на достигнутом. Собственно, Вас обязательно должно заинтересовать не только то, как выводить данные на экран, но и как оперировать этими данными. Например, производить какие-либо вычисления. Безруких жонглеров не бывает, и, пока часть шаров находится в воздухе, оставшуюся часть циркач держит в руках. Что бы что-либо хранить (в частности данные) необходимо иметь хранилище. Для нашей программы таким хранилищем будет оперативная память. Прежде чем что-то где-то разместить, необходимо подобрать подходящую упаковку. Скажем, Вы вряд ли станете наливать молоко в спичечный коробок. В программировании, перед тем, как разместить информацию в

оперативной памяти, Вы обязательно должны определить характер этой информации. Итак, **Типы данных**.

**Тип данных** — понятие, определяющее максимальный размер (в байтах) и тип информации, которая будет использоваться программой.

Программирование отчасти отражает объекты внешнего мира, изрядно их упрощая. В начале изучения мы столкнемся с вещами, с которыми по сути Вы сталкивались много раз. Давайте условно разделим все типы данных на следующие группы:

1. Числовые.
2. Символьные.
3. Логические.

Далее мы рассмотрим ряд ключевых слов, используемых в языке C для обозначения типов данных.

### **Числовые типы.**

Числа, как известно, бывают целые и вещественные. Вещественные числа мы будем называть **числа с плавающей точкой**. Особо отметим, что запятая, отделяющая целую часть от дробной, меняется на точку. Например 7,8 в C записывается 7.8

Переменные, в которых мы будем хранить значения вещественных чисел, будут объявляться типа **float** или **double**. В чем разница между этими типами? Тип float описывает числа с плавающей точкой одинарной точности, а double — двойной. Поясним, что в математике точность определяется количеством цифр, которые представляют число. Двойной точностью называют метод представления чисел с удвоенным, по сравнению с обычным, количеством цифр. Вот характеристики типов для чисел с плавающей точкой:

Пояснение	Тип	Размер в байтах
описывает вещественные числа одинарной точности	float	4
описывает вещественные числа двойной точности	double	8

Кроме вещественных в C предусмотрено три типа объявляющих целочисленные данные. В таблице приведены основные характеристики этих типов:

Пояснение	Тип	Размер в байтах	Диапазон значений
описывает целые числа	int	4	от -2147483648 до 2147483647
описывает короткие целые числа	short	2	от -32768 до 32767
описывает длинные целые числа	long	4	от -2147483648 до 2147483647
описывает длинные целые числа	long long	8	от -9,223,372,036,854,775,808 до 9,223,372,036,854,775,807

### Символьный тип.

Тип предназначен для хранения только одного символа. Сразу же предупредим — типа для хранения строк в C не существует.

Пояснение	Тип	Размер в байтах
описывает символы	char	1

## Логический тип.

Тип предназначен для хранения логических данных. Подробнее мы познакомимся с ним позже. Логические данные могут принимать одно из двух значений: истина (true) либо ложь (false).

Пояснение	Тип	Размер в байтах	Значения
описывает логические значения	bool	1	true false

**Примечание:** Если необходимо исключить из диапазона типа данных отрицательные значения, перед названием типа следует указать ключевое слово `unsigned`. Например, `unsigned int`. Такой тип будет включать в себя только положительные значения от 0 до 4294967294.

Итак, мы выяснили, какие бывают типы данных и какие ключевые слова языка C используются для их обозначения. В заключении, следует отметить, что язык C является регистрозависимым (т. е. ЗАГЛАВНЫЕ и строчные буквы в нем это не одно и то же). Обратите внимания на то, что все выше описанные типы данных записаны строчными буквами. Следите за этим, т. к. `int` — это тип данных, а `INT` — ошибка.

В следующей теме мы рассмотрим применение типов данных на практике.



## 6. Переменные и константы

Мы с Вами уже познакомились с типами данных и знаем как классифицируется информация для хранения. Осталось выяснить, как данные записать в оперативную память и как к ним потом обращаться, чтобы использовать или изменить.

Итак, меняющиеся данные договоримся называть ПЕРЕМЕННЫМИ, а постоянные данные — КОНСТАНТАМИ.

**Переменная** — область оперативной памяти, обладающая собственным именем и предназначенная для хранения данных, которые могут быть изменены.

**Константа** — область оперативной памяти, обладающая собственным именем и предназначенная для хранения постоянных данных.

Вот пример констант: Всем известное количество дней в неделе и количество месяцев в году... Оно не меняется ни при каких обстоятельствах, — поэтому эти значения-константы.

А вот наш возраст — величина переменная. Сегодня кому-то 26 лет, а через год будет 27.

Из определений становятся понятно, что для поиска данных в памяти им дают имена (по аналогии с тем, что вещи в багажном вагоне снабжают бирочками). В среде программирования их называют идентификаторами. Одна из первых проблем, которую решают родители новорожденного - это выбор имени для него. Накладывает ли имя

отпечаток на характер человека, на его судьбу — вопрос сложный и спорный. Можно услышать совершенно противоположные мнения на этот счет. Но тот факт, что имя (идентификатор), даваемое новой переменной (константе), должно быть интуитивно понятным и объясняющим назначение переменной, не будет оспаривать ни один более-менее опытный программист.

Имена данным даются, соблюдая строго определенные правила. Этих правил нарушать нельзя!

### **Правила составления имен.**

В имени допустимо использование только следующих символов:

1. Прописные и строчные буквы латинского алфавита. При этом не забывайте о регистрозависимости языка. Например, Age и age — это два разных имени.

2. Цифры. Однако, цифра не может быть использована в качестве первого символа. То есть, Name1 допустимо, 1Name — нет.

3. Символ подчеркивания «\_». Дело в том, что вы должны помнить, что пробел, тоже является символом и данный символ недопустим в имени переменной. Его заменит знак подчеркивания, который улучшит выразительность имен. Например, сравните: ageofman и Age\_Of\_Man.

При определении имени для переменной запомните следующее:

1. Нельзя называть переменную ключевыми словами языка программирования. Ключевое слово — слово, зарезервированное под синтаксис языка программирования (int, float, double и т. п.). В Visual Studio ключевые

слова подсвечиваются синим светом, это как минимум приведет к путанице.

2. Нежелательно существование двух идентификаторов с одинаковыми именами.

3. Нельзя использовать никакие другие символы, кроме допустимых. (см. выше)

## **Объявление и использование переменных и констант.**

Теперь мы обладаем всей информацией для создания (объявления) переменной. Осталось лишь выяснить, каков общий синтаксис:

1. **тип\_данных имя\_переменной**; — в данном случае в оперативной памяти будет выделена ячейка размером, соответствующим заданному типу. И этой ячейке будет присвоено выбранное Вами имя. Что же там будет содержаться? В только что созданную переменную будет записано случайное число, определяемое операционной системой. Это число будет содержаться в памяти до тех пор, пока Вы не заполните переменную другим значением, с помощью специального оператора присваивания =

2. **тип\_данных имя\_переменной=значение**; — существует и такая возможность — заполнить переменную значением, прямо при создании. Такой процесс мы будем называть инициализацией.

3. **const тип\_данных имя\_переменной=значение**; — а это объявление константы. Основные моменты состоят в том, что вне зависимости от типа данных перед ним указывается ключевое слово **const**. Кроме того константа обязательно должна быть инициализирована при создании. Поменять ее значение впоследствии будет невозможно.

## Показ значения переменной на экран.

Показ значения переменной на экран осуществляется с помощью `cout<<`

```
cout<<имя_переменной; // кавычки в данном случае не указывают
```

Можно показывать содержимое нескольких переменных через `<<`

```
cout<<имя_переменной1<<имя_переменной2;  
// кавычки в данном случае не указывают
```

Можно чередовать показ содержимого переменных с текстовыми сообщениями и Escape-последовательностями через `<<`

```
cout<<"Текст"<<имя_переменной1  
<<"Текст"<<имя_переменной2<<"\n";
```

Показ содержимого констант осуществляется по полной аналогии с переменными.

## Практические примеры

Приведем несколько примеров создания и инициализации переменных и констант для разных типов данных.

## Целочисленные переменные и константы.

С целыми числами мы встречаемся повсеместно: возраст, количество стульев, количество комнат, количество дней в неделе и т.д.

Переменные, в которых будут храниться целые числа, ОБЪЯВЛЯЮТСЯ так:

```
int Age;
```

О чем говорит эта строчка? Что в переменной по имени Age (возраст) будет храниться целое значение. Слово `int` объявляет ТИП значения переменной по имени Age.

Теперь например, мы хотим внести в переменную Age значение 34. Как это сделать?

```
Age =34;
```

Эта строчка читается так: «Переменной Age присвоить значение 34».

Еще раз посмотрим на оператор присваивания: `Age =34;`

Слева от знака равно стоит имя переменной, которой присваивается значение. А справа стоит то значение, которое присваивается.

Константа, в которой будет храниться целое число объявляется так:

```
const int Count_Days_in_Week=7;
```

О чем говорит эта строка? Слово `const` подчеркивает, что объявляется константа. `int` сообщает, что константа будет целым числом. Затем следует имя константы `Count_Days_in_Week` и ее значение 7.

Теперь разберем, как вычислять значение переменной. Для чего это нужно? Простой пример: как посчитать сколько часов в 2000 году? Неужели вы хотите посчитать эту число сами?

На самом деле, довольно легко заставить компьютер это сделать самостоятельно. От нас требуется написать только формулу этого вычисления.

В 2000 году 366 суток, в сутках 24 часа. Значит формула расчета количества часов в 2000 году такова: 366 умножить на 24.

В языке С в качестве знака умножения используют \* (звездочка, комбинация Shift+8).

Разработаем программу, которая считает сколько же часов в 2000 году.

Перед созданием программы рекомендуется кратко набросать ее алгоритм.

Алгоритм — последовательность действий, направленная на решение поставленной задачи.

Дано: количество дней в году — 366. Это значение не будет меняться, поэтому объявим его константой целого типа по имени `DayIn_2000Year`. количество часов в сутках — 24. Тоже не меняется. Объявим ее константой целого типа по имени `HourInDay`. В нашей программе будет единственная переменная, в нее мы запишем результат расчета. Назовем эту переменную `HourIn_Year2000`. Она будет целого типа. (`int`)

Алгоритм будет следующий:

1. Объявление и инициализация переменных и констант.
2. Подсчет результата.
3. Вывод на экран результата.

Имена переменным вы можете придумать сами (не забывайте только о правилах составления имен переменных).

А теперь — как всегда, создадим новый проект и введем следующий код:

```
// Заголовок
#include <iostream>
// определение пространства имен, в котором есть cout<<
using namespace std;
// Главная функция
void main()
{
    // вывод пустой строки
    cout<<"\n";
    //Объявляем целочисленные константы
    int DayIn_2000Year=366;
    int HourInDay=24;
    //объявляем целочисленную переменную
    int HourIn_Year2000;
    // вычисляем искомое значение и
    // помещаем его в переменную HourIn_Year2000
    HourIn_Year2000=DayIn_2000Year*HourInDay;
    // выводим значение переменной HourIn_Year2000 на экран
    cout<<"\t\t In 2000 year "<< HourIn_Year2000;
    cout<<" hours\n ";
}
```

Все! Компилируйте программу!

## Вещественные переменные и константы.

Пример объявления и инициализации

```
float Weight;
Weight=12.3452;
double weight_atom;
weight_atom= 2.5194e+017;
```

Что обозначает число 2.5194e+017?

Это краткая запись вещественных чисел. Называется она — экспоненциальной формой записи чисел. Сообщаем вам секрет расшифровки написанного. Этим набором символов описывается число 251940000000000000 или  $2,1594 \times 10^{17}$ .

$-3.4E008$  расшифровывается так:  $3,4 \times 10^{-8}$ , что аналогично  $3,4:108$ .

$-1.5E+003$  расшифровывается как  $-1,5 \times 103$ .

Числа с плавающей точкой типа `float` могут меняться от  $-3,4 \times 10^{38}$  до  $3,4 \times 10^{38}$ .

Значения от  $-3,4 \times 10^{-38}$  до  $3,4 \times 10^{-38}$  считаются равными нулю.

А теперь давайте поработаем с вещественными числами на практике:

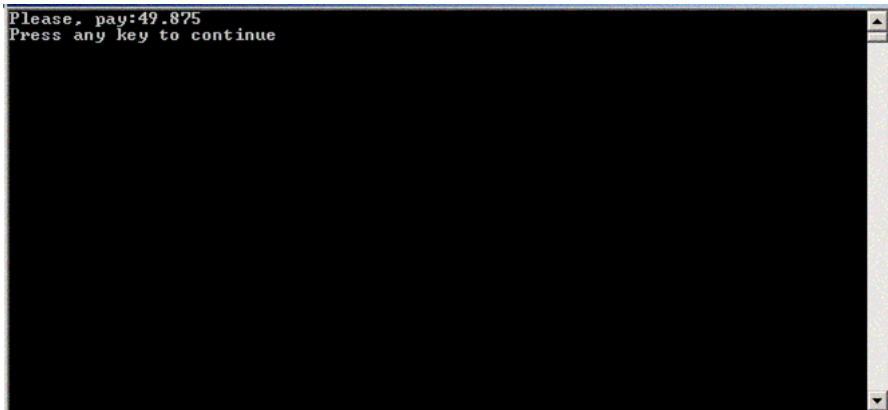
Напишем программу, которая будет рассчитывать стоимость покупки. Пусть программа использует цену товара (`Cost`), количество купленного товара (`Count`), и, учитывая скидку (`Discount`), вычисляет стоимость покупки (`Price`).

Создадим новый проект **Pokupka** и введем текст следующей программы

```
// Заголовок
#include <iostream>
// определение пространства имен, в котором есть cout<<
using namespace std;
// Главная функция
void main()
{
    //Объявляем переменную Discount
    float Discount=0.05;
    //Объявляем переменную Cost
    float Cost=10.50;
    //Объявляем переменную Count
    int Count=5;
    //Объявляем переменную Price
    float Price;
    //Вычисляем значение переменной Price
    Price=Count*Cost-Count*Cost*Discount;
    // Выводим итоговую стоимость товара со скидкой
    cout<<"Please, pay:"<<Price<<"\n";
}
```



Компилируйте программу и отправляйте ее на выполнение. То, что Вы должны увидеть на экране, приведено ниже.



```
Please, pay:49.875
Press any key to continue
```

## Символьные и логические переменные и константы.

В данном уроке мы не будем приводить примеры использования символьных и логических переменных и констант. Их предназначение более подробно будет описано в будущем. Оговорим лишь объявление и инициализацию.

```
// Логическая переменная
bool Flag;
Flag=true;
// Один символ всегда указывается в одинарных кавычках
char Symbol='A';
/* Escape - последовательность рассматривается компилятором, как один символ
   и соответственно может быть записана в переменную или константу
   типа char*/
const char NewLine='\n';
cout<<NewLine// показывает пустую строку
```

## 7. Ввод данных

Вы уже знакомы с операцией вывода информации на экран компьютера — `cout`, но в большинстве программ требуется не только выводить какую-либо информацию на экран, но и иметь возможность ввести в компьютер какие-либо данные с клавиатуры. В предыдущем разделе была приведена программа расчета скидки. Естественно, что такие параметры как цена и количество товара было бы неплохо ввести с клавиатуры на этапе выполнения программы. Давайте рассмотрим как Вы это можете сделать.

Если нам нужно ввести данные в компьютер, то будем пользоваться командой `cin`. Как ею пользоваться? Синтаксис оператора ввода:

```
cin>>имя_переменной;
```

`имя_переменной` указывает на переменную, в которую нужно поместить данные, введенные с клавиатуры:

**Например:**

```
cin>>Age;
```

Эта команда помещает число, введенное с клавиатуры, в переменную с именем `Age`. Для того, чтобы ввести число в переменную `Number`, нужно всего лишь набрать такую вот команду:

```
cin>>Number;
```

Ввод сразу нескольких переменных, записывают таким образом:

```
cin>>имя_переменной1>>имя_переменной2>>...>>имя_переменнойN;
```

Список имен переменных должен содержать имена всех переменных, в которые Вы хотите ввести данные с клавиатуры. Список имен может состоять из любого количества имен переменных, разделенных комбинацией символов >>.

```
cin>>Quantity>>Price>>Discount;
```

Давайте добавим в программу Рокирка ввод данных с клавиатуры:

```
//Заголовок
#include <iostream>
// определение пространства имен, в котором есть cout<<
using namespace std;
// Главная функция
void main()
{
    //Объявляем переменную Discount
    float Discount=0.05;
    //Объявляем переменную Cost
    float Cost=10.50;
    //Приглашение ввести цену товара
    cout<<"What's the cost?\n";
    //Ввод значения в переменную Cost
    cin>>Cost;
    //Объявляем переменную Count
    int Count=5;
    //Приглашение ввести количество
    cout<<"How much?\n";
    // Ввод значения в переменную Count
    cin>>Count;
    //Объявляем переменную Price
    float Price;
    //Вычисляем значение переменной Price
```

```
Price=Count*Cost-Count*Cost*Discount;
// Выводим итоговую стоимость товара со скидкой
cout<<"Please, pay:"<<Price<<"\n";
}
```

Теперь Вы увидели особенность работы оператора `cin>>`. Как только программа встречает данный оператор, она останавливается и ждет реакции пользователя. И, пока пользователь не введет данные и не нажмет "Ввод"(Enter). Только после этого продолжится выполнение.

На примере еще раз поработаем с вводом и выводом. Напишем программу-обманщика: программа предлагает поиграть в числа, кто загадает большее число тот и выигрывает.

Создадим новый проект **Game** и введем такой текст:

```
// Заголовок
#include <iostream>
// определение пространства имен, в котором есть cout<<
using namespace std;
// Главная функция
void main()
{
    // Приглашение "Давай играть!"
    cout<<"Let's play!\n";
    //Объявление переменной i
    int i;
    //Приглашение "Введите число"
    cout<<"Enter a number:";
    //Ввод числа
    cin>>i;
    //Вывод числа, которое "загадал" компьютер
    cout<<"I have "<<i+1<<"\n";
    // Вывод результата игры
    cout<<"I'm winner!\n";
}
```

Откомпилируйте программу. Обращаться с нашей программой легко. Просто вводите любое число и постоянно оказывается, что у компьютера число больше и он выигрывает. Вот, что Вы увидите на экране при запуске программы, если на запрос "Enter a number:" введете число 67:

```
Let's play!
Enter a number: 67
I have 68
I'm winner!
Press any key to continue...
```

Почему он все время выигрывает?  
Давайте рассмотрим строчку

```
cout<<"I have "<<i+1<<"\n";
```

В ней выводится значение переменной *i*, значение которой Вы ввели с клавиатуры, увеличенное на 1, то есть компьютер всегда выводит число, на 1 больше введенного вами с клавиатуры.

Если в этой команде заменить выражение *i+1* на выражение *i-1*, то выигрывать всегда будете вы, так как число выводимое компьютером всегда будет на единицу меньше введенного вами с клавиатуры.

В заключение хотим обратить Ваше внимание на операторы + (плюс) и - (минус). Они используются для сложения и вычитания. В языке C, также есть оператор для деления /. Данная информация поможет Вам при выполнении домашнего задания, а более подробно мы поговорим об операторах в следующих уроках.

## 8. Литералы

Литералы (literals) — это фиксированные значения, которые программа не в состоянии изменять. Для каждого типа языка C существуют литералы, включая символьный и булевский типы, целые числа и числа с плавающей точкой. Как это не парадоксально, типа данных для хранения строк в C не существует, а строковые литералы существуют.

### Некоторые примеры:

5	целочисленный литерал-int
5l	l или L означает long
true	логический литерал-bool
5.0	литерал с плавающей точкой, понимается как double
5.0f	f или F— с плавающей точкой, понимается как float
0.3e-2	литерал с плавающей точкой double, e или E отделяют экспоненциальную часть
'd'	символьный литерал
"Visual"	строковый литерал — это набор произвольных символов, заключенный в кавычки. Компилятор воспринимает его именно как набор символов и никак обрабатывать его не собирается, даже если в кавычках окажутся какие-то ключевые слова и операции.

### Примеры кода, содержащего литералы.

```
// "abracadabra" - строковый литерал '\n' - символьный литерал
cout<<"abracadabra"<<"\n";
int a = 2; // 2 - литерал типа int
```

На этом изложение материала первого урока можно считать завершенным. Настоятельно рекомендуем Вам выполнить домашнее задание, описанное в следующем разделе. Желаем удачи!

## 9. Домашнее задание

---

1. Напишите программу, вычисляющую среднее арифметическое двух чисел.
2. Напишите программу, которая переводит гривны в \$, €, российские рубли.
3. Выведите на экран следующий текст:  
"To be or not to be"  
    \Shakespeare\  
4. В С нет операции возведения в квадрат. Напишите программу, которая вычисляет квадрат любого, введенного числа.
5. Введите три числа и выведите на экран значение суммы и произведения этих чисел.

















