STEP computer ACADEMY

PROGRAMMING **C**

Lesson No. 1

Introduction to Programming

Contents

1. Prolegomena

Opening Address

Welcome to the world of programming! What is programming? Perhaps, each of you once heard this word. To paraphrase a famous quote, we can say: "Programmer: this sounds good!" And so it is. If you came across job-offering websites when surfing the Internet, you would probably pay attention to the payment that was offered to programmers. Certainly, the following question arises: why is pay level for programmers so high? All happens according to the laws of the market: there is a demand for programmers, but the number of sufficient specialists is too small at the moment. Of course, if one could become a good programmer in a short period of time, just having learnt a number of terms, the profession would not be so popular. But do not worry! We, your instructors, will try to do everything we can in order to not only tell you how to code, but also teach you to live programming. However, our co-operation must be mutual. The instructor, no matter how much he tries, will not be able to teach the student, if the latter does not want it. Only constant self-cultivation will help you master every aspects of programming.

We sincerely hope that you are fond of the magical world of programming!

Before proceeding to the study of any science, it is always necessary to find out where this science appeared, how it developed, what are its roots and meaning in history.

We sincerely hope that you are fond of the magical world of programming!

Before proceeding to the study of any science, it is always necessary to find out where this science appeared, how it developed, what are its roots and meaning in history.

Historical Facts

The 19th-century English mathematician Shanks spent more than 20 years of his life to calculate pi accurate within 707 significant digits after the point. That result got the glory of the calculation record in the 19th century. However, later it was discovered that Shanks made a mistake in the 520th digit and, therefore, all subsequent significant digits were calculated incorrectly.

In 1804, the French inventor Joseph Marie Jacquard created a "program-controlled" weaver's loom. Punched cards, interconnected in the form of a tape, were used to control the loom. Guided by the holes in the punched cards arranged in different order, wooden studs of the machine "reading device" determined which threads were to lift, and which threads were to lower in order to obtain the desired pattern.

In 1890, the US inventor Herman Hollerith developed an electromechanical computing machine-tabulator controlled with punched cards. It was used for tabulation of the results on the US population census. A tabulator manufacturer founded by Hollerith later turned into the corporation International Business Machines (IBM).

In 1936, 25-year old student from Cambridge University, an Englishman Alan Turing published a paper "On computable numbers" which considered a hypothetical device ("Turing machine"), suitable for solving any solvable mathematical or logical problem — a prototype of a programmable computer.

In 1941, the German engineer Konrad Zuse built an operational computer Z3, which was using binary system. Programs were recorded on a punched tape.

In 1945, the Higher Technical School, University of Pennsylvania (USA), the physicist John Mauchly and the engineer Prosper Eckert built fully electronic machine "ENIAC". To program a machine, it was necessary to manually install thousands of switches and plug hundreds of sockets in the contact panel plugs.

June 1, 1945, the Hungarian-born American mathematician John von Neumann circulated a report called "The First Draft of a Report on the EDVAC", which contained the concept of storing computer instructions in its own internal memory.

June 21, 1948, the University of Manchester (UK), the first in the world memory-stored program focused on the search of the largest factor of the current digit was performed on the machine "Mark-1".

In 1949, the computer "EDSAC" was designed under the leadership of Maurice Wilkes. The designers who worked on EDSAC introduced a system of mnemonic notations, where each machine instruction was represented by a capital letter, and automated subroutine setup to a specific location in the memory. Maurice Wilkes called the mnemonic scheme and subroutine library an assembly system — hence, the word "assembly" has been common.

In 1949 in Philadelphia (USA), "Short Code" — the first primitive programming language interpreter — was created under the direction of John Mauchly.

In 1951 in the company Remington Rand, the American programmer Grace Hopper developed the first translator. Hopper called it a compiler.

In 1957, on the 20th floor of the IBM headquarters on Madison Avenue in New York, the language Fortran (FORmula TRANslation) was born. 30-year-old mathematician John Backus led the team of developers. Fortran is the first "real" high-level language.

In 1963, the programming language BASIC was created. The founders of the language were John Kemeny and Thomas Kurt, researchers at the Dartmouth College. Under the direction of the founders, the language was implemented by a group of college students. The first dialect is known as Dartmouth BASIC.

1958–1968 were the years of development and improvement of the programming language called Algol, whose name came from the word-combination "algorithmic language". Unlike Fortran, mainly used in the US and Canada, Algol was widely distributed in Europe and the USSR. The language was created by an international committee, which included European and US scientists — John Backus, Peter Naur, Niklaus Wirth.

In 1970, Niklaus Wirth created a programming language and named it in honor of the French physicist and mathematician Blaise Pascal. Wirth planned to use Pascal as a language teaching procedural programming.

In 1972, 31-year-old system programming specialist from the company Bell Labs, Dennis Ritchie developed the C programming language.

The first description of the language was given in the book written by B. Kernighan and D. Ritchie. This description was the standard for quite a long time, but some aspects were unclear, which entailed many interpretations of C. To change that situation, the Standardization Committee for C was formed at the American National Standards Institute (ANSI).

In 1983, the standard of the language C known as ANSI C was approved.

In the early 1980s, in the same Bell Laboratory, Bjarne Stroustrup created an essentially new language called "C with Classes", as a result of additions and extensions of C.

In 1983, the name was changed to C ++.

May 23, 1995, the company Sun Microsystems released a new programming language called Oak. The language was designed for programming consumer electronics. Later Oak was renamed Java language and began to be widely used in the development of applications and server software.

In 2000-2001, a new programming language C # (C-Sharp), specially designed for the platform .NET, was adopted and standardized. Employees of the Microsoft Research (a scientific research institute attached to the corporation Microsoft) — Anders Hejlsberg, Scott Viltamut, Peter Golde and other well-known experts, including Eric Meyer — were involved in the creation of the language. The language version C # 2.0 was presented in spring 2005. It should be noted that one of the languages, on which basis C # was created, was the same good old C ++.

So, from a short list of historical facts, we have found that C and C ++ are two different programming language, even though C ++ is based on C. Our classes will be devoted to the study of these two programming languages. Of course, we will obtain new knowledge in a consistent manner and explore the language C first, and then we will gradually move on to C ++.

After we made a little trip to the past, we can move on to the present. We will definitely need software for our further work. With the rapid development of the market of IT-technologies, newer versions of various programs have been constantly created. In the following sections of the lesson, we will look at the installation of the software, which we will use as a part of our learning process.

2. Installing Microsoft Visual Studio 2013

Microsoft Visual Studio is a package of software products from the company Microsoft, which is used for the development of solutions of various scales: from students' lab researches to corporate level projects. Visual Studio includes an integrated development environment (IDE), numerous tools and utilities. Using Visual Studio, you can create console applications as well as applications with complex graphical interface. We will use products from the Visual Studio product line, as a part of our learning process.

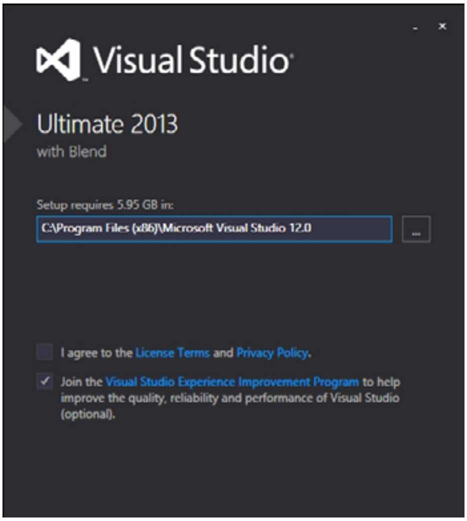
History of Visual Studio began in 1997, the year when Visual Studio 97 was released. You will learn programming using Microsoft Visual Studio 2013. This is the current version at the moment. Before we consider the installation of Visual Studio, let us analyze an issue related to the editions of Visual Studio. Visual Studio edition is a version of Visual Studio, which differs from other edition with its set of features. Visual Studio 2013 has the following editions: Ultimate, Premium, Professional, Test Professional, and Express Editions.

Ultimate is the most complete edition of Visual Studio. Each edition has its own pricing policy. The only exception is Express Editions. This is a set of Visual Studio (s), "sharpened" to meet specific needs, for example, for the development of mobile solutions, web-based solutions, or desktop solutions, etc. You can use Express Editions absolutely free, including commercial development. It is important to note that different

Visual Studio editions are available in different languages. For example, you may want to install the Cambodian localization. Don't do this! As a part of your future professional activity, you will meet colleagues and projects from other countries, which certainly will use the English interface. That is why you should always prefer the language of the original version, that is English.

We will consider the installation of Visual Studio by the example of Microsoft Visual Studio 2013 Ultimate and Microsoft Visual Studio Express 2013 for Windows Desktop. Let us start with Ultimate.

To start installing Ultimate, insert the DVD disc into the DVD-ROM drive (or mount the ISO-image of the DVD disc, using programs such as Daemon Tools Lite), run the file vs_ultimate.exe and you will see the following screen:



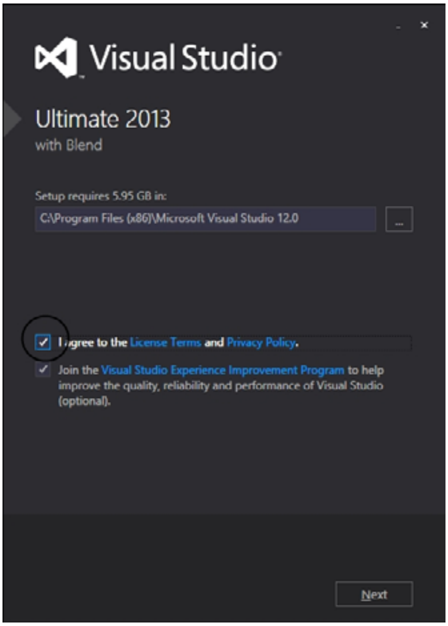
Setup requires 5.95 GB in:
C:\Program Files (x86)\Microsoft Visual Studio 12.0

I agree to the License Terms and Privacy Policy.

✓ Join the Visual Studio Experience Improvement Program to help improve the quality, reliability and performance of Visual Studio (optional).

In this window you can select the path for the Visual Studio installation and decide whether you want to participate in the program dedicated to Visual Studio enhancement. If you want to participate in the program designed for quality improvement, leave the checkmark in the option "Join the Visual Studio Experience Improvement Program ...." To start the Visual Studio installation, you need to tick the point "I agree to the License Terms and Privacy Policy".

By selecting this option, you acknowledge thereby that you have read and agreed to all points of the license. If you have made the right choice, the button «Next» will appear at the bottom of the window. Click on it. After that you will see a window that allows you to select additional software to install. At this point, you can leave the default products. At a later time you will be able to install components. To continue the process, click on the button "INSTALL"

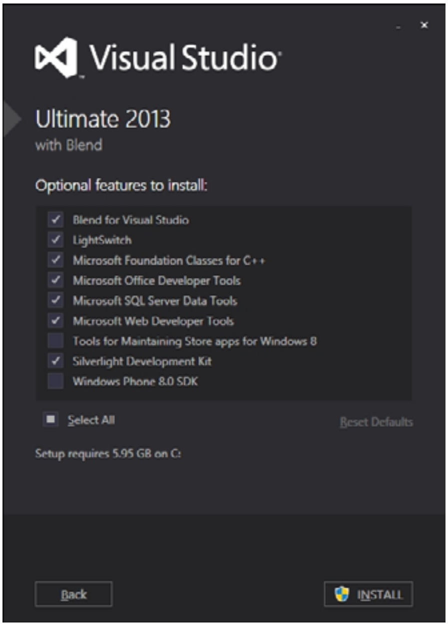

Visual Studio

Ultimate 2013
with Blend

Setup requires 5.95 GB in:
C:\Program Files (x86)\Microsoft Visual Studio 12.0

✓ I agree to the License Terms and Privacy Policy.

✓ Join the Visual Studio Experience Improvement Program to help improve the quality, reliability and performance of Visual Studio (optional).


Visual Studio

Ultimate 2013
with Blend

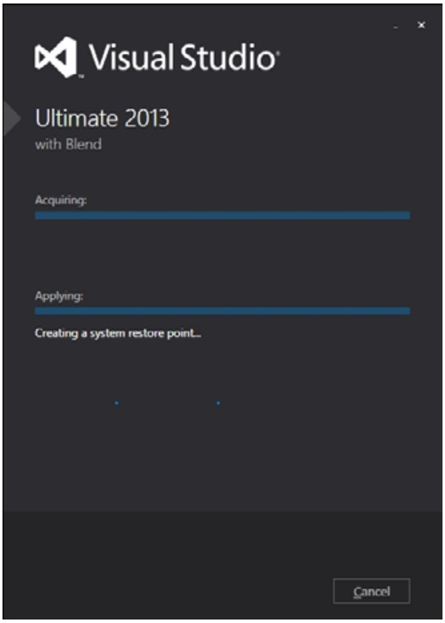
Optional features to install:

- ✓ Blend for Visual Studio
- ✓ LightSwitch
- ✓ Microsoft Foundation Classes for C++
- ✓ Microsoft Office Developer Tools
- ✓ Microsoft SQL Server Data Tools
- ✓ Microsoft Web Developer Tools
- ☐ Tools for Maintaining Store apps for Windows 8
- ✓ Silverlight Development Kit
- ☐ Windows Phone 8.0 SDK

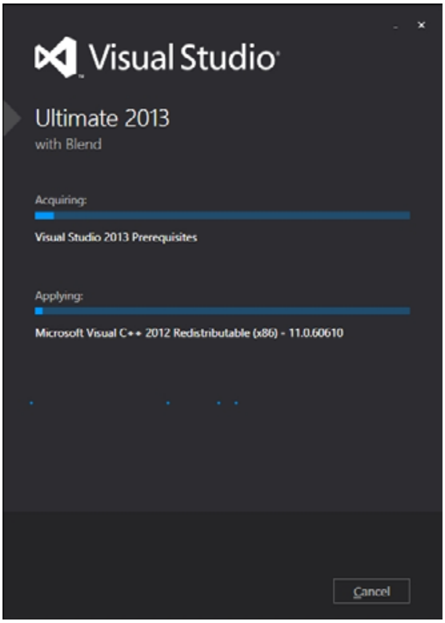
■ Select All

Setup requires 5.95 GB on C:

The installation process starts from this moment. It will take some time, during which you will be able to watch the progress in the setup. Some windows typical for this state are provided below.



Creating a system restore point...

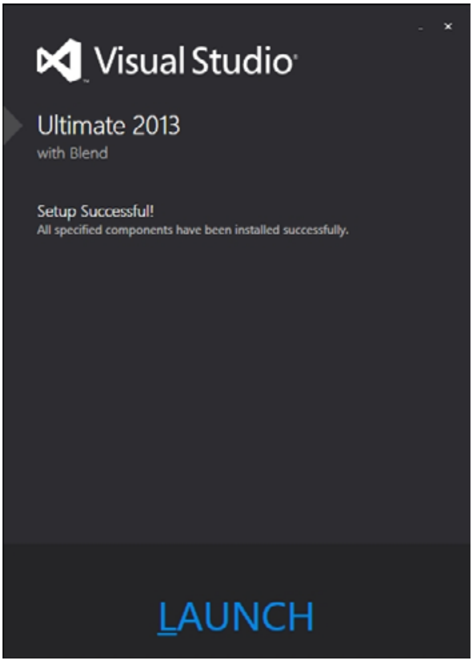


Visual Studio 2013 Prerequisites

Microsoft Visual C++ 2012 Redistributable (x86) - 11.0.60610

After installation, if everything has been completed successfully, you will see a window with the button "LAUNCH" in it. By clicking on it, you can run Visual Studio for the first time or use a familiar interface to run Windows programs (Start menu, shortcuts, etc.)

Run Visual Studio to ensure that your previous actions were successful. During the first start, you may be asked to enter your Microsoft Live ID. At the same time you have the opportunity not to enter it. To do this, select the point "Not now, may be later".



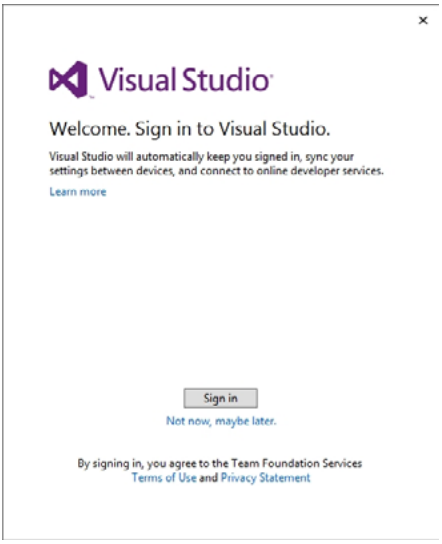
Visual Studio

Ultimate 2013

with Blend

Setup Successful!

All specified components have been installed successfully.

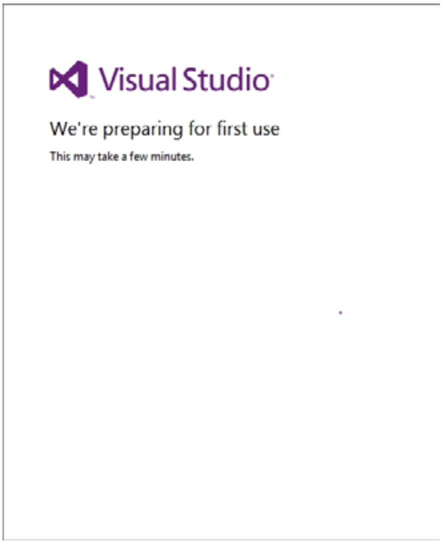


Visual Studio

Welcome. Sign in to Visual Studio.

Visual Studio will automatically keep you signed in, sync your settings between devices, and connect to online developer services.

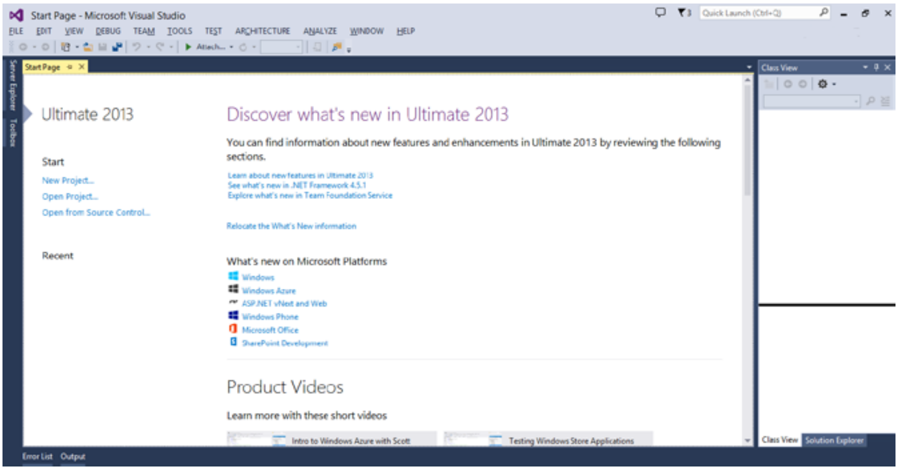
By signing in, you agree to the Team Foundation Services Terms of Use and Privacy Statement



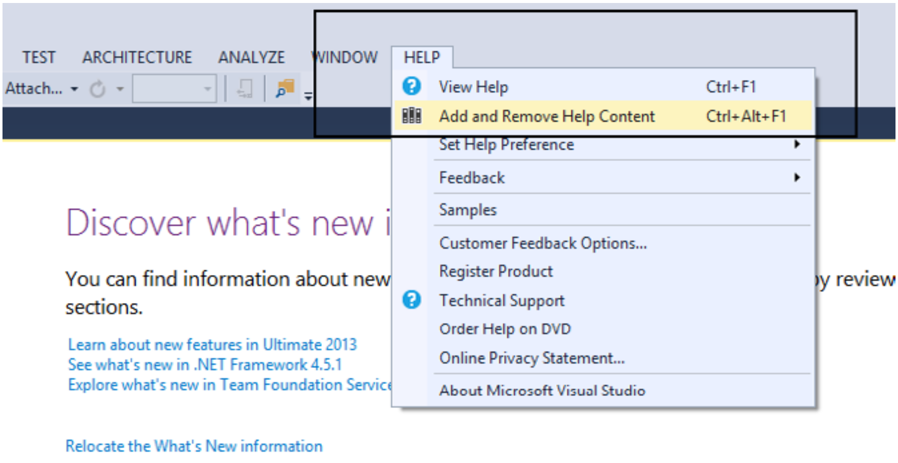
Visual Studio

We're preparing for first use

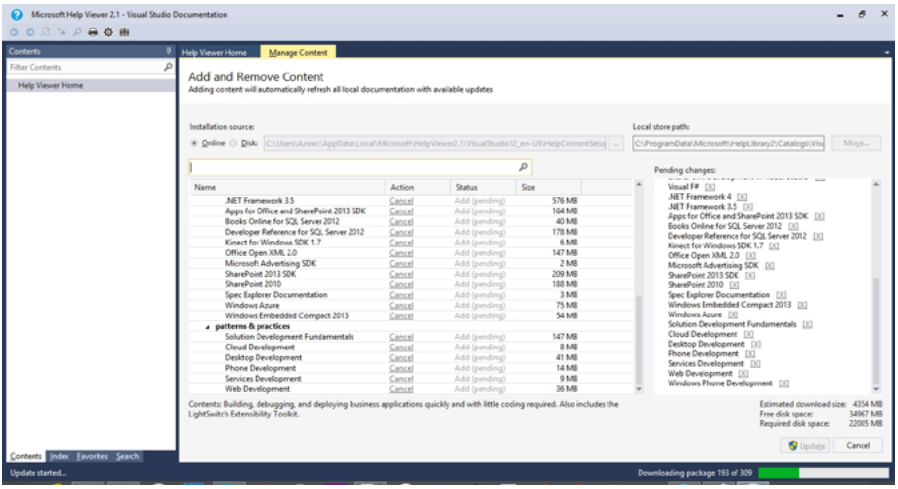
This may take a few minutes.


Start Page - Microsoft Visual Studio
FILE EDIT VIEW DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Quick Launch (Ctrl+Q)
Start Page
Ultimate 2013
Start
New Project...
Open Project...
Open from Source Control...
Recent
Discover what's new in Ultimate 2013
You can find information about new features and enhancements in Ultimate 2013 by reviewing the following sections.
Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service
Relocate the What's New information
What's new on Microsoft Platforms
Windows
Windows Azure
ASP.NET vNext and Web
Windows Phone
Microsoft Office
SharePoint Development
Product Videos
Learn more with these short videos
Intro to Windows Azure with Scott
Testing Windows Store Applications
Class View
Solution Explorer
Error List
Output

When being trained and developing applications, the background information (MSDN) from Microsoft will be rather helpful for you. To install a local copy of the reference, you need to select the menu item ***Help-> Add and Remove Help Content or press Ctrl + Alt + F1***.


TEST
ARCHITECTURE
ANALYZE
WINDOW
HELP
Attach...
View Help Ctrl+F1
Add and Remove Help Content Ctrl+Alt+F1
Set Help Preference
Feedback
Samples
Customer Feedback Options...
Register Product
Technical Support
Order Help on DVD
Online Privacy Statement...
About Microsoft Visual Studio
Discover what's new i
You can find information about new
by review
sections.
Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service
Relocate the What's New information

You will see a panel with a menu of reference material. We recommend choosing the link "Add" everywhere. Help is never too much :) After material selection, click "Update"for iyd local installation.



Add and Remove Content
Adding content will automatically refresh all local documentation with available updates

Installation source:
Online Disk: C:\Users\Anexc\AppData\Local\Microsoft\HelpViewer2.1\VisualStudio12_en-US\HelpContentSetu

Local store path:
C:\ProgramData\Microsoft\HelpLibrary2\Catalogs\Visu

Name	Action	Status	Size
.NET Framework 3.5	Cancel	Add (pending)	576 MB
Apps for Office and SharePoint 2013 SDK	Cancel	Add (pending)	164 MB
Books Online for SQL Server 2012	Cancel	Add (pending)	40 MB
Developer Reference for SQL Server 2012	Cancel	Add (pending)	178 MB
Kinect for Windows SDK 1.7	Cancel	Add (pending)	6 MB
Office Open XML 2.0	Cancel	Add (pending)	147 MB
Microsoft Advertising SDK	Cancel	Add (pending)	2 MB
SharePoint 2013 SDK	Cancel	Add (pending)	209 MB
SharePoint 2010	Cancel	Add (pending)	188 MB
Spec Explorer Documentation	Cancel	Add (pending)	3 MB
Windows Azure	Cancel	Add (pending)	75 MB
Windows Embedded Compact 2013	Cancel	Add (pending)	54 MB
patterns & practices			
Solution Development Fundamentals	Cancel	Add (pending)	147 MB
Cloud Development	Cancel	Add (pending)	8 MB
Desktop Development	Cancel	Add (pending)	41 MB
Phone Development	Cancel	Add (pending)	14 MB
Services Development	Cancel	Add (pending)	9 MB
Web Development	Cancel	Add (pending)	36 MB

Contents: Building, debugging, and deploying business applications quickly and with little coding required. Also includes the LightSwitch Extensibility Toolkit.

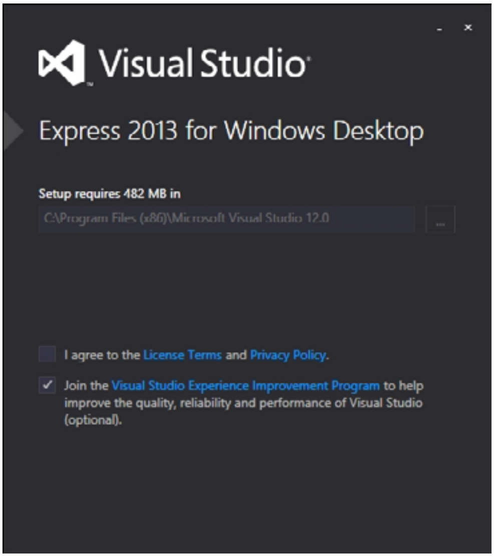
Pending changes:
Visual F# [X]
.NET Framework 4 [X]
.NET Framework 3.5 [X]
Apps for Office and SharePoint 2013 SDK [X]
Books Online for SQL Server 2012 [X]
Developer Reference for SQL Server 2012 [X]
Kinect for Windows SDK 1.7 [X]
Office Open XML 2.0 [X]
Microsoft Advertising SDK [X]
SharePoint 2013 SDK [X]
SharePoint 2010 [X]
Spec Explorer Documentation [X]
Windows Embedded Compact 2013 [X]
Windows Azure [X]
Solution Development Fundamentals [X]
Cloud Development [X]
Desktop Development [X]
Phone Development [X]
Services Development [X]
Web Development [X]
Windows Phone Development [X]

Estimated download size:	4354 MB
Free disk space:	34967 MB
Required disk space:	22005 MB

Update started...
Downloading package 193 of 309

Now we consider the process of installing Microsoft Visual Studio Express 2013 for Windows Desktop. It is very similar to the installation of Ultimate.

To start installing Express, insert the DVD disc into the DVD-ROM drive (or mount the ISO-image of the DVD disc using program such as Daemon Tools Lite) and run the file ***wdexpress_full.exe***, after which you will see the following window:


Visual Studio

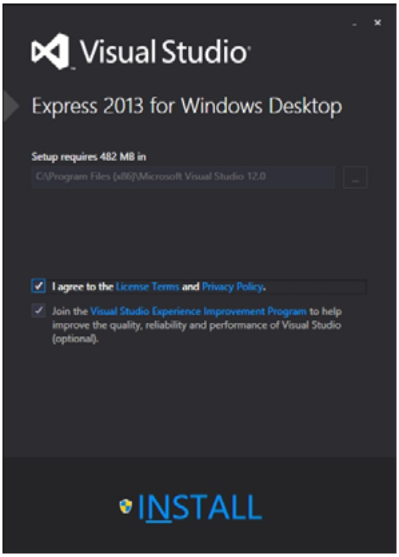
Express 2013 for Windows Desktop

Setup requires 482 MB in

C:\Program Files (x86)\Microsoft Visual Studio 12.0

☐ I agree to the License Terms and Privacy Policy.

✓ Join the Visual Studio Experience Improvement Program to help improve the quality, reliability and performance of Visual Studio (optional).


Visual Studio

Express 2013 for Windows Desktop

Setup requires 482 MB in

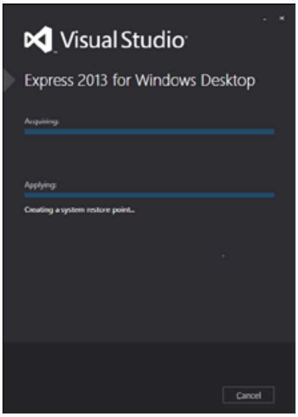
C:\Program Files (x86)\Microsoft Visual Studio 12.0

✓ I agree to the License Terms and Privacy Policy.

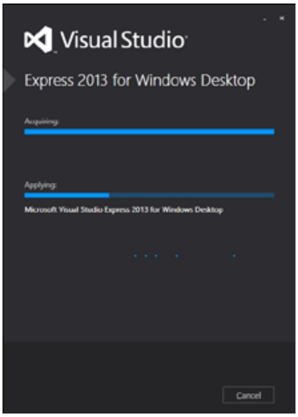
✓ Join the Visual Studio Experience Improvement Program to help improve the quality, reliability and performance of Visual Studio (optional).

Again, in this window you can select the path for the installation of Visual Studio and decide whether you want to participate in a program dedicated to Visual Studio enhancement. If you want to participate in the program designed to improve the quality of the installation procedure of the product, leave the checkmark in the option "Join the Visual Studio Experience Improvement Program...." To start the installation of Visual Studio, you will need to select the point "I agree to the License Terms and Privacy Policy".

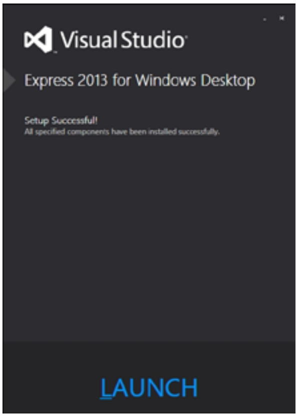
By selecting this option, you acknowledge thereby that you have read and agreed to all points of the license. If you have made the right choice, the button "INSTALL" will appear at the bottom. Click on it and the process of installing Visual Studio will begin.



Creating a system restore point...



Microsoft Visual Studio Express 2013 for Windows Desktop



Setup Successful!
All specified components have been installed successfully.

The installation process starts from this moment. It will take some time, during which you will be able to watch the progress in the setup. Some windows typical for this state are indicated below.

After installation, if everything has been completed successfully, you will see a window with the button "LAUNCH" in it. By clicking on it, you can run Visual Studio for the first time or use a familiar interface to run Windows programs (Start menu, shortcuts, etc.) As with the installation of the Ultimate edition, you will be asked to enter your LIVE ID (you can ignore this point). Don't forget to install the help system.

If you have passed all points of the installation successfully, you are ready to write your first program.

That is all. Now you can move on to the following sections of the lesson, where we will learn to code.

3. Installing Microsoft Visual Studio 2015

New versions of Visual Studio are appearing as the years are passing by. You already know how to install and configure Visual Studio 2013. Now we will talk a little bit about what Microsoft Visual Studio 2015 is. As you have probably guessed, it is a newer version of Visual Studio, released to the public in 2015. Certainly, the new version of the software is always better and more powerful than the old one. Visual Studio 2015 always follows this rule.

There are three versions of Visual Studio: Visual Studio Community 2015, Visual Studio Professional 2015, Visual Studio Enterprise 2015.

Let us talk about each of them.

- ***Visual Studio Community 2015*** is a free version available for download to all developers. It has rich features and can be used for any purpose, both commercial and non-commercial. This is a good choice to start your acquaintance with this product.
- ***Visual Studio Professional 2015*** is a paid version of the product. It contains a large number of enhancements as compared to the free version.
- ***Visual Studio Enterprise 2015*** is a full paid version of the product. It provides powerful functionality that is needed in the development of the most complex enterprise applications.

A detailed table focusing on the comparison of the versions is available here.

The installation process as well as the process of the project creation in any ***Visual Studio versions 2015*** is similar to Visual Studio 2013. To work on our course, you can install Visual Studio Community 2015.

4. The First Project

A friend of mine once told me about his impressions from visiting Prague:

"If you meet a waiter, who does not speak Russian, then take the menu and point your finger on the things you want to order. Sometimes similar words mean different things. For example, the word "fruit" sounds almost like "vegetables" in Czech. Guess what will the waiter bring you, if you ask him for a vegetable salad? ☺".

That friend of mine wanted to warn his friends about a possible mistake, but he would be hardly able to help, if we had to communicate with the computer instead of a waiter. Alas, the latter does not know at what menu bar you are poking your finger. The computer thoroughly follows clear instructions. These instructions must be given by means of special commands. The commands, in turn, make up the whole independent languages. Languages understood by the computer are called programming languages. Conclusion: in order to find a common language with Czech waiter, one has to be a ready-witted person. To find a common language with the computer, one must know the language.

From the first section, you already know that C is a programming language. The language that allows us to clearly explain to the computer what we want from it, though, to be perfectly frank, the computer understands only one language, i.e. the language of machine codes. For example, a program that displays the phrase "Hello, world!", in its "native" computer language looks roughly like:

```
_YH¦ў+._5lЧ№+f-H¦ў+ +f+ ¤д-ы+fO_fO¦¤+эm№¤+¤+¦Ф№+¤-;
-wGГ=xФ№+MZP __ + @ A -¦ ¦ -!+ L-!T
u  ._5lЧ№+f-&ЛZ¶QW&бфЬ№+&Л &ЛJf&OB_Ф№
+f;sЛlФ№+Л&lt; _t=ИФ№+_G¶u_O¶WfM+-чfЛ№3+M-Ac°
_>] Л¦Л+ЛМЛm _сбшЬ№+Л 9ШФ№+t§Л§pФ№+ш№) fd¦ __+RPбшЬ
№+Л 9ШФ№+tЛ§pФ№+шC) +  +Э
tAc°-щ3-Л_Ф№+f;s §lФ№+Л +u"бАФ№+¤+MAc°ЎD At  Э+- 3+
ыўы•ы°ыЎыЇЛD$П$._4Ч№++     ._ dЛ4 Ў- ubSЛ dґ4 +__є
--fУЙf-+
-+ЙA+@У№+Л-[Xf +tRPRVh•ў+ш_S Z+єfO-fЛцf-d¤+§ RfRfh ш4
Л+Zы¦SшC¤д-ы¦¤+ьM-¤++¤_e fM+fO_fO¦f¤бЛЁdЎ  t3+OшAc°Л§AФ№+-
dб4 ЙdЙ§4 f_f _t dЛ4 Ў- uSSЛ dґ4 +__є --fУЙf-+-+ЙA+@У№+Л-[
+єЙ]№fM+fO-fЛц+ыf-d¤+§ RfRfh шз Л+Zы+Sш¦
$ PE L  ¦7 p ¤!  ` P  oK   P ў+       0
```

You will object and state that it is impossible to write programs using such code and you will be absolutely right! However, we will not use it. A programming language is required for it in order to facilitate programming. Programming languages are divided into two main groups: INTERPRETED and COMPILED. This division relates to the specialized software that translates commands from the programming language into the machine language — the COMPILER or INTERPRETER. Let us find out what is the difference between them. Imagine that we have a file that contains a set of commands.

The First Situation. Commands are Written in an Interpreted Language

Every time you start the program, the interpreter checks the code line by line. If no syntax errors are found, the commands are converted to the machine code (a set of instructions for the processor). The program will be launched for execution. If there is an error, the interpreter will stop and you will be prompted to correct it and run the program again. However, even if there are no more errors and you have already written the program, the interpreter will be triggered every time you run the program in order to verify the code again. Thus, we can conclude that the machine version of the code is not stored anywhere. Disadvantages of this approach are as follows: code performance is reduced, but the check is impossible to disable.

The Second Situation. Commands are Written in a Compiled Language

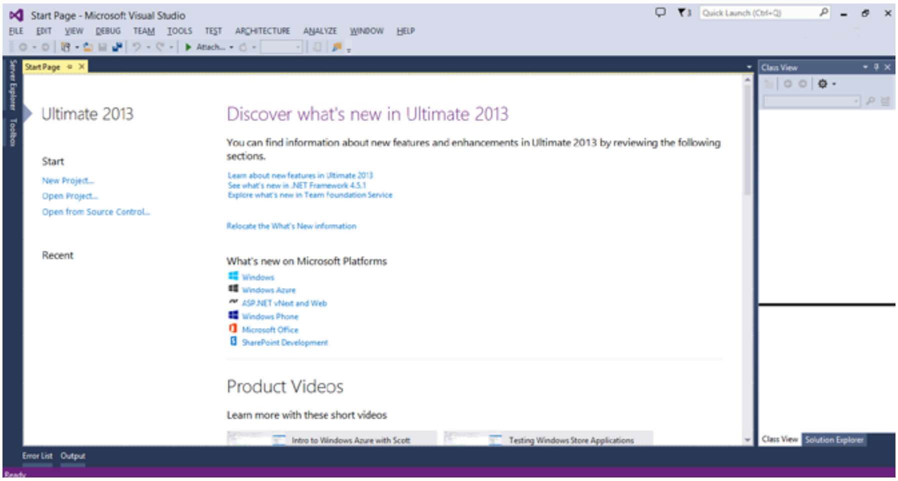
The compiler operates almost as the interpreter, i.e. checks the code line by line. If it encounters an error, it does not stop but continues examining the code to the end, revealing all subsequent errors and generating error messages. Besides it, the compiler generates a special object file with an extension .OBJ. This file contains the text of the program translated into the machine language. However, the computer does not work directly with the file. There is a concept of arrangement or linking. A linker is another special program, which collects the machine code (from the file with the extension .OBJ) and various auxiliary data into a single executable file with the extension .EXE. This file can be run for execution as a separate, stand-alone program and the compiler is not involved in its launching.

The C language, to the study of which you proceed, is a compiled language. In our case, when working with the Microsoft Visual Studio Compiler shell, call to the compiler is performed automatically and it allows to translate C commands in the machine code, so to say, "with a subtle movement of your hand".

The First Attempt at Writing

One of the founders of C, Brian Kernighan, once said: "The only way to learn a new programming language is to write programs on it". That is what we are going to do now. As is customary in the world of programming, the first program written on a new language is the program "Hello world!".

To write your first program, launch the program shortcut Microsoft Visual Studio 2013 from the menu "Start" -> "All Programs" -> "Microsoft Visual Studio 2013" or in some other way familiar to you. After the launch, you will see an image similar to that shown in the figure:


Start Page - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
FILE EDIT VIEW DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Attach...
Start Page
Server Explorer
Toolbox
Class View
Ultimate 2013
Start
New Project...
Open Project...
Open from Source Control...
Recent
Discover what's new in Ultimate 2013

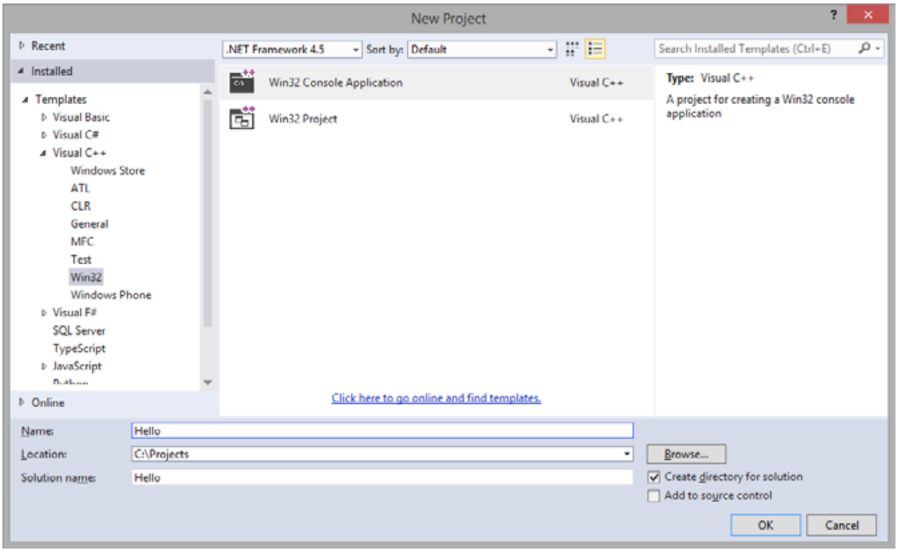
You can find information about new features and enhancements in Ultimate 2013 by reviewing the following sections.

Learn about new features in Ultimate 2013
See what's new in .NET Framework 4.5.1
Explore what's new in Team Foundation Service
Relocate the What's New information
What's new on Microsoft Platforms
Windows
Windows Azure
ASP.NET vNext and Web
Windows Phone
Microsoft Office
SharePoint Development
Product Videos
Learn more with these short videos
Intro to Windows Azure with Scott
Testing Windows Store Applications
Class View
Solution Explorer
Error List
Output
Ready

Now we try to create a project that will, represent our program eventually. We will explore our project in detail further, when we write large programs. Until we think of the project as an association of several files. Now let us do it step by step:

- Once you have downloaded the shell, select the menu item ***File -> New -> Project***. You will see a dialog box.

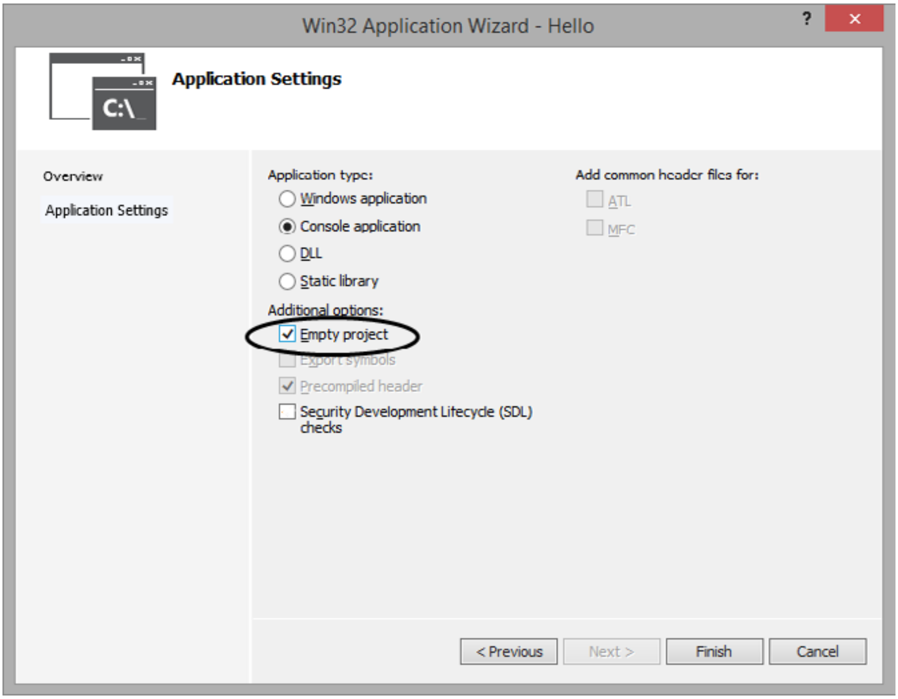
- In the dialog box New Project on the list Installed Templates select the project type ***Visual C ++ -> Win32 -> Win32 Console Application***
- In the field Location, select the disk and folder to contain our project. To do this, enter C: \ Projects (or any other name of the folder convenient for you where you will store all your homework projects) in this field.
- Name the PROJECT — to do it, enter a project name Hello in the field Name.
- Now you can press the OK button.



Type: Visual C++
A project for creating a Win32 console application

Name: Hello
Location: C:\Projects
Solution name: Hello
[✓] Create directory for solution
[ ] Add to source control

Now you can see the configuration window of the project properties — select the tab Application Settings. Check the field ***Empty Project*** — it means that we are creating an empty project. Uncheck the field ***Security Development Lifecycle (SDL) checks***. Now press the Finish button.


Win32 Application Wizard - Hello

Application Settings

Overview
Application Settings

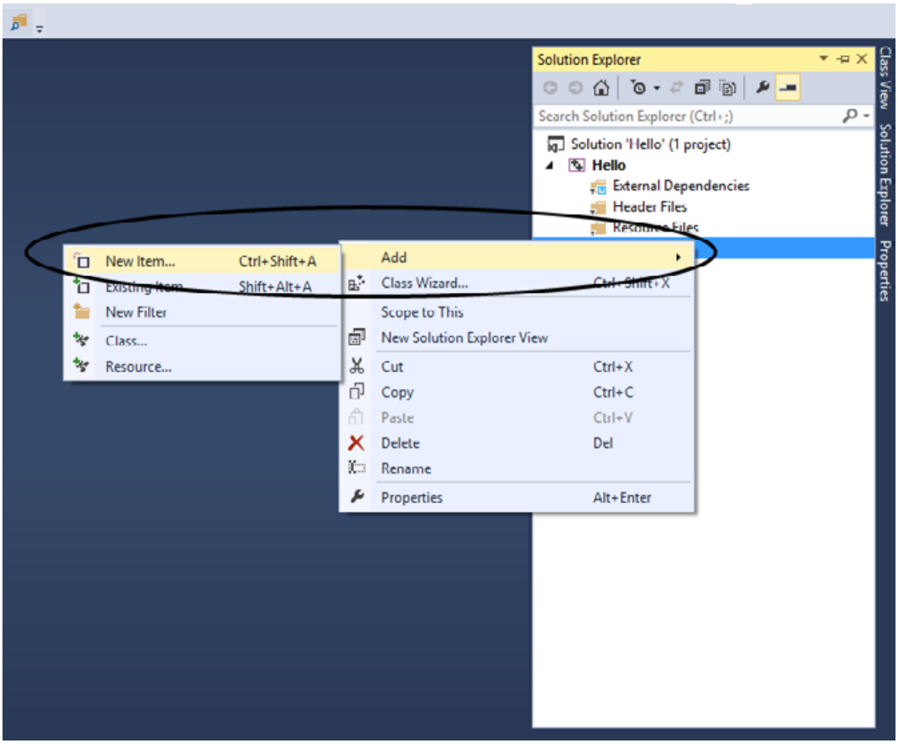
Application type:
- Windows application
- [✓ Console application]
- DLL
- Static library

Additional options:
- [✓ Empty project]
- Export symbols
- [✓ Precompiled header]
- Security Development Lifecycle (SDL) checks

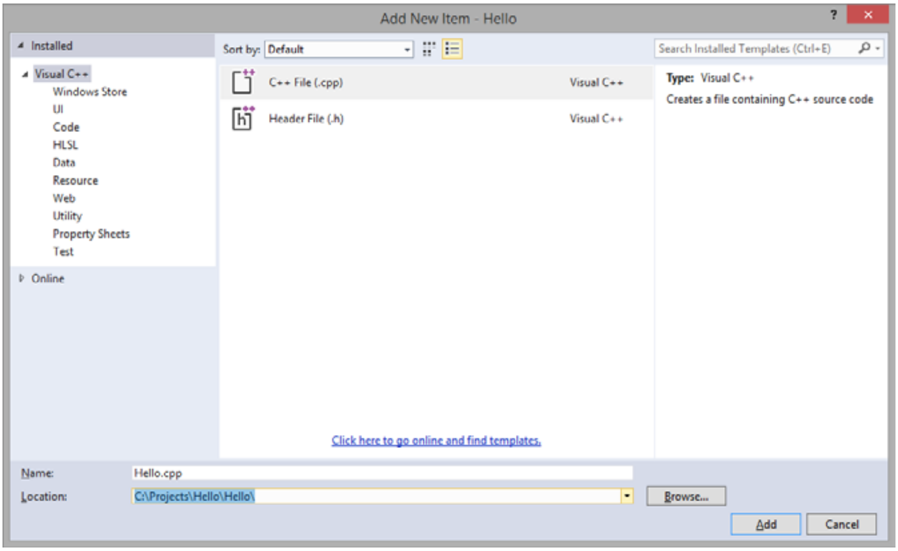
Add common header files for:
- ATL
- MFC

So, we have prepared a space to locate our program. In order not to stop halfway, add an empty file to the project. We will use it to type the text of our program. To do this, follow these steps:

- On the right, there is a window titled Solution Explorer. To call this window, you can also use the keyboard shortcut ***Ctrl + Alt + L***. In this window, right-click on the folder named Source Files.



- In the drop-down menu, select ***Add-> Add New Item*** ...
- The file selection window is opened. Again, you have a huge selection. We recommend to select the icon C ++ File (.cpp) (a file containing a program written in C / C ++).
- In the text field Name (file name), enter the file name Hello. Click Add.


Add New Item - Hello
Installed
Visual C++
Windows Store
UI
Code
HLSL
Data
Resource
Web
Utility
Property Sheets
Test
Online
Sort by: Default
C++ File (.cpp) Visual C++
Header File (.h) Visual C++
Search Installed Templates (Ctrl+E)
Type: Visual C++
Creates a file containing C++ source code
Click here to go online and find templates.
Name: Hello.cpp
Location: C:\Projects\Hello\Hello\
Browse...
Add
Cancel

After clicking on the Add button, you will have a text area, where you can write your first program.

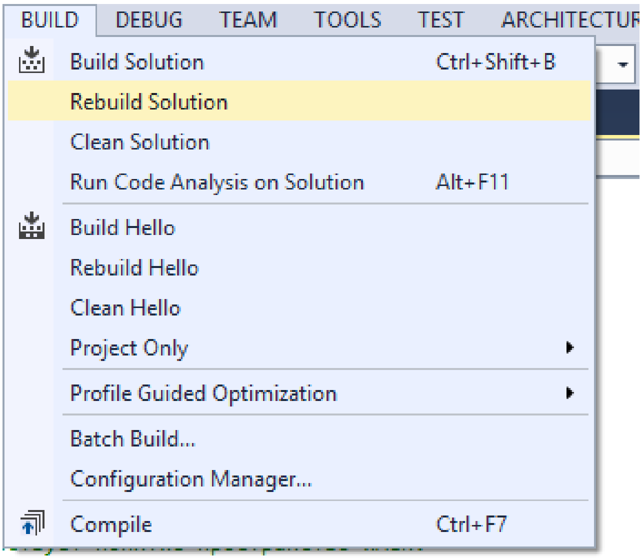
An Example of Your First Program in C

Before you begin to write your first program, let us introduce the concept of a comment, for convenience. Comments mean program notes, which are intended solely for the programmer. The compiler ignores them. For example, the comment designates the use of one or other line of the program. The example below shows how to work with comments.

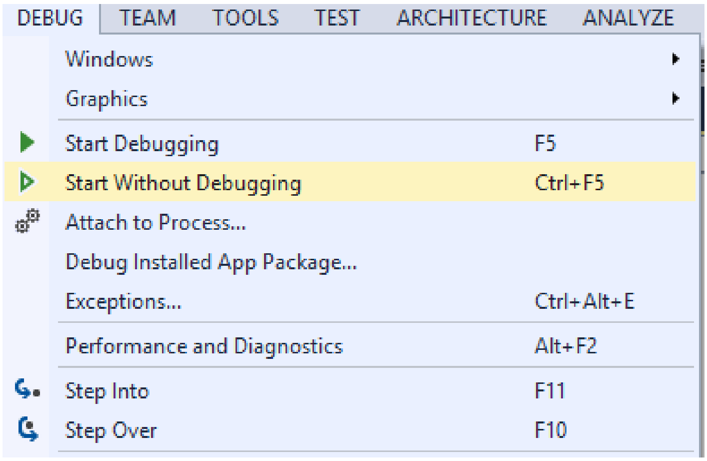
Now, in the appeared text box, type the following cod:

```
//These are comments
//They are green
//Comments start with two slashes //
/* If you want to create a multi-string comment
   used the construction */
/* a comment */
/* This line connects the library
   iostream to the program. The library is a file that
   contains the description of various functions
   implemented by other programmers.
   The program can use the functions contained in
   the library iostream now
 */

#include <iostream>
/* In C ++, there is a concept of namespaces.
   Such space defines a certain area, at which actions
   of the statement or function are aimed.
   In order to use the statement located in
   the certain namespace, you need to connect this
   namespace in your program.
   The namespace called std
   is connected below using namespace std;
 */

void main () // Start of the program, herefrom
             // the program will start its execution
// The entire text of the program is placed between
// the braces
{        // This is a brace
         // The next line displays the greeting
         // Hello, World!
         // This action is carried out using cout <<
         // The library and the namespace, in which
         // it is located were linked for its operation
    cout << "Hello, World! \ n";
         // put a semicolon in the end of
         // the command. This sign must END
         // each command in C.
}
```

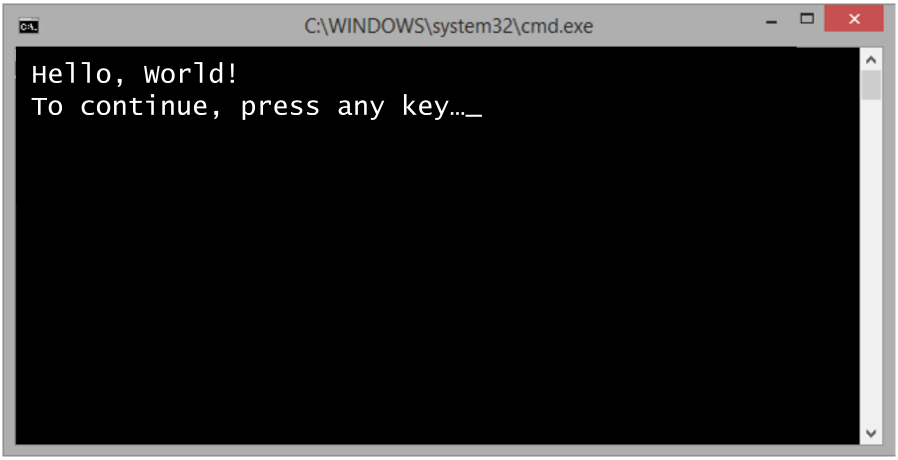
As you already know, the computer understands only the language of machine codes. So before the program is executed by the computer, it must be translated into a language of machine codes. As you remember, the compiler will perform it, not we. Projects written in C contain a lot of files. Let us compile all files included in the project at once. To do it, select the menu item Build in the menu bar, then select Rebuild Solution (rebuild all)



We hope you have typed the text without mistakes. Our program has been successfully translated into the language of machine code and you can run it for execution now. Running the program is a very simple procedure. In the Debug menu, select Start Without Debugging.



The program will work out and you will see a window here:

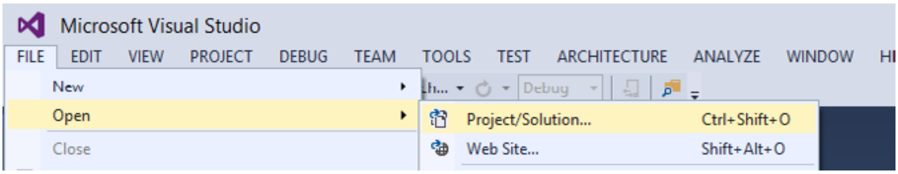


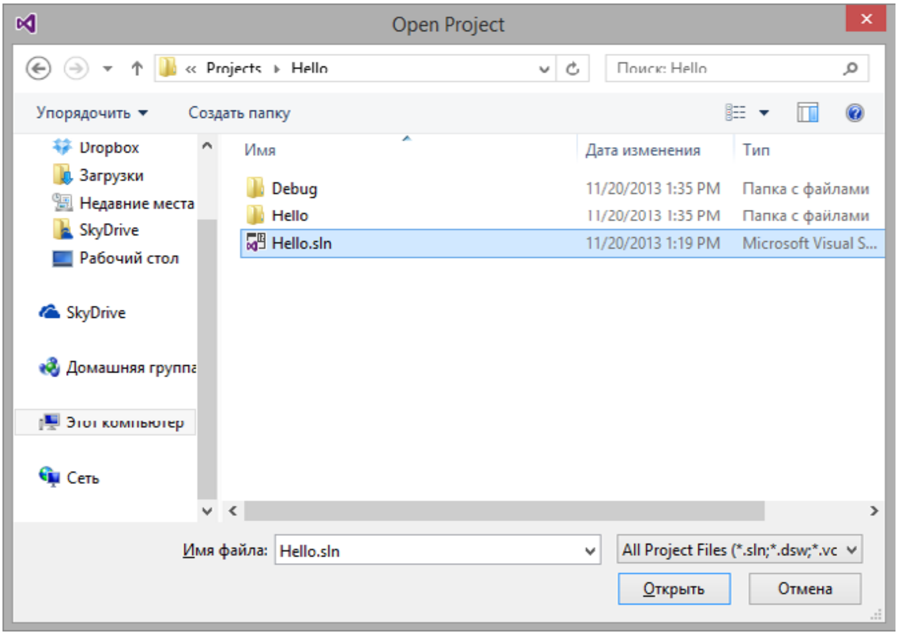
```
Hello, world!
To continue, press any key..._
```

Congratulations! It was pretty easy, isn't it? To complete the program, press any key on the keyboard.

Opening a Saved Project

To restore a previously saved project on the disk, you should run Visual Studio (if you have not launched it yet). In the menu File, select the item ***Open -> Project / Solution*** and specify the name of your project.


Microsoft Visual Studio
FILE EDIT VIEW PROJECT DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW
New
Open
Close
Project/Solution... Ctrl+Shift+O
Web Site... Shift+Alt+O
Debug



Once the project is opened, you can continue working on it. You can find projects created as a part of the lesson in the corresponding subfolder called ***Sources***. In particular, you can find this project located on the path ***lesson folder/ Sources/Hello***.

5. Data Output

You already know that we can display various text lines using the command cout <<.

However, to make the compiler understand this command, you must be aware of the following three main points:

1. The title of the program should contain a line #include <iostream>.
2. Before using the command, you must connect the namespace, to which the command cout belongs.

```
using namespace std;
```

3. We write the line that we want to display using the command cout << in quotation marks. For example:

```
cout<<"Here we write what we want";
```

The command cout << not only displays lines but also allows to arrange them. To arrange the line output, special control characters, which makes a combination of the character \ and the character that defines the action to make over the line are used. These control characters are called ***Escape*** sequences. Here are some of them:

```
\b    Delete the last displayed character
\n    Go to the beginning of a new line
\t    Go to the next tab stop
\\    Display the backslash  \
\"    Display the double quotes  "
\'    Display the single quote  '
```

The existence of the last three Escape sequences is always slightly confusing first. Why should we use control characters, if we can simply write "or \ or '? The answer is above ground: all these three characters are statements and if they are "simply written", the compiler will take them as statements. For example, when a word is used figuratively, it is enclosed in quotes. Suppose you need to display the following text on the screen:

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

If you do not use Escape sequences, it is obvious that your command will look like this:

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

This will lead to the inevitable error. The compiler will accept only a part of the line, namely cout << "The Man in red was". It will take double quotes after was as closing and it will take everything else as an incorrect syntax. Such program, of course, will not start running for execution. The correct version is as follows:

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

Now, let us discuss where Escape sequences should be indicated in cout <<. The most important thing that you need to know is that Escape sequence must always be inside the quotation marks because it is a text, while your further possibilities are virtually unlimited. For example:

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

As a result of this command, we will see the following output on the screen:

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

A Practical Example of Using Cout <<

We will write a program that displays a brief summary about the studied Escape sequences. We want to see on the screen:

```
\b   Backspace
\n   New line
\t   Horizontal tab
\\   Backslash \
\"   Double quotation mark "
\'   Single quotation mark '
```

Run the environment ***Visual Studio 2013***. Create a new project under the name ***EscapeSequences***. Type the code indicated below.

```
// Title
#include <iostream>
// definition of a namespace, which contains cout<<
using namespace std;
// Main function
void main()
{
/* The following command displays the text in 4 tabs
   Escape Sequences
      and transfers the output to the next line  */
                  cout<<"\t\t\t\tEscape Sequences\n";
```

```
              // Prints a blank line
                                             cout<<"\n";
/* In 2 tabs it displays the text \b,
   and displays Backspace in another 1 tab
   Then \n moves the output to the next line */
                         cout<<"\t\t\\b"<<"\tBackspace\n";
             // Prints a blank line
                                             cout<<"\n";
/* In 2 tabs it displays the text \n,
   and in another 1 tab New line
   Then \n moves the output to the next line */
                          cout<<"\t\t\\n"<<"\tNew line\n";
            // Prints a blank line
                                             cout<<"\n";
/* n 2 tabs it displays the tex \t,
   and in another 1 tab displays Horizontal tab
   Then \n moves the output to the next line */
                    cout<<"\t\t\\t"<<"\tHorizontal tab\n";
              // Prints a blank line
                                             cout<<"\n";
/* n 2 tabs it displays the tex \\,
   and in another 1 tab displays Backslash \
   Then \n moves the output to the next line */
                     cout<<"\t\t\\\\"<<"\tBackslash \\\n";
              // Prints a blank line
                                             cout<<"\n";
/* n 2 tabs it displays the tex \t,
   и еще через 1 табуляцию Double quotation mark "
   Then \n moves the output to the next line */
      cout<<"\t\t\""<<"\tDouble quotation mark \"\n";
             // Prints a blank line
                                             cout<<"\n";
/* n 2 tabs it displays the tex \',
  and in another 1 tab Double quotation mark  '
   Then \n moves the output to the next line */
      cout<<"\t\t\'"<<"\tSingle quotation mark \'\n";
```

```
                   /// Prints a blank line
                                                            cout<<"\n";
}
```

Compile the program (***Build -> Rebuild Solution***). If you have a lot of errors, remember the following rules:

If the program will print the message on the screen, then the line #include <iostream> is written in the beginning of the program and the namespace, to which the command cout (using namespace std;) belongs, is connected.

Each program must contain a function called main (). The program starts with this function.

Commands of the function main () are located inside the curly braces {}

All commands must end with a semicolon.

Run it (***Debug -> Start Without Debugging***).

P. S.

You must have noticed that we use only Latin characters when displaying data. This is due to the fact that we have written our program in Microsoft Windows, and it is carried out in MS DOS. The fact is that each character in any operating system has a numerical code. The system identifies this character precisely with this code. Cyrillic character code in MS DOS and Windows is not the same, so the program will not work correctly when using the Cyrillic alphabet[1]. For example, we wrote under Windows:

```
cout<<"Morning";
```

[1] The indicated example is valid for the Cyrillic alphabet. Note that inscription depend on the system language on your computer.

A appears on the screen:

```
µЄЁю
```

This can be easily explained by the fact that the letter O is 238 under Windows, for example, and it corresponds to the letter "ю" according to the DOS code. Latin code is the same in both operating systems. Later we will learn how to correct this situation.

In the standard C ++ language 11 programmers are provided a new opportunity to display special characters on the screen. To do this, use the "raw" line. If the line is declared as "raw", the compiler will interpret it character by character. To declare a line "raw", use the following format.

```
R"(line_text)"
```

R indicates that it is a "raw" line. The contents of the line must be enclosed in brackets. Here are some examples:

```
cout<<R"(hello\nworld)"; // on the screen  hello\
nworld
   cout<<R"("Test 'string'\t")"; // on the screen
"Test 'string'\t"
   cout<<R"((Such brackets))"; // on the screen
(Such brackets)
```

6. Types of Data

After you have read the previous sections of the lesson, writing a program that displays anything on the screen does not cost anything to you.

```
// Title
#include <iostream>
// definition of the namespace, which contains cout<<
using namespace std;
// Main function
void main()
{
    // Print the phrase "Congratulations on a good start!"
    // Phrase is displayed in 3 tabs,
    // two blank lines are added
    cout<<"\t\t\tCongratulation with good beginning!!!
    :)\n\n";
}
```

That is actually all that you can do for now (yet).

A man should never stop halfway. Actually, you have to be interested in not only how to display data on the screen, but also how to handle the data to make any calculations, for example. Handless jugglers just do not exist, and while a part of the balls is in the air, the performer is holding the rest. To store something (data, in particular), you need to have a repository. RAM will be a repository for our program. Before placing something somewhere, you need to find the right package. Say, you will unlikely pour the milk into the matchbox. In programming, before you place the information in the main memory, you must determine the nature of the information. Thus, we consider ***data types.***

Data Type is a concept, which determines the maximum size (in bytes) and type of information to be used by the program.

Programming partly reflects the objects of the outer world, considerably simplifying them. At the beginning of the study we will face things, which, in fact, we have been facing a lot of time. Let us conditionally divide all data types into the following groups:

1. Numeric.
2. Character.
3. Logical.

Next we will look at some keywords used in the C language to describe data types.

Numeric types

Numbers, as you may know, can be integers and real. We will call real numbers **the *floating-point numbers***. We note in particular that the comma separating the integer part from the fractional part is changed to the point. For example, 7, 8 is written as 7.8 in C.

The variables, in which we will store the values of real numbers, will be announced of the types ***float*** or ***double***. What is the difference between these two types? The type float describes single-precision floating-point numbers and the type double describes double-precision floating-point numbers. We will explain that accuracy in math is determined by the number of digits that represent the number. Double precision is a method of representing numbers with doubled, compared to the usual, number of digits. Here are the characteristics of types with floating-point numbers:

Desription	Type	Size in bytes
Describes single-precision real numbers	float	4
Describes double-precision real numbers	double	8

In addition to real numbers, C provides three types declaring integer data. The table shows the main characteristics of these types:

Desription	Type	Size in bytes	Range of values
Describes integers	int	4	from -2147483648 to 2147483647
Describes short integers	short	2	from -32768 to 32767
Describes long integers	long	4	from -2147483648 to 2147483647
Describes long integers	long long	8	from -9.223.372.036.854.775.808 to 9.223.372.036.854.775.807

Character type

The type is intended for storing only one character. We will immediately warn you — there is no type intended for storing lines in C.

Description	Type	Size in bytes
Describes characters	char	1

Logical Type

The type is intended to store logical data. We will look at it in detail later. Logical data can take one of two values: ***true or false***.

Description	Type	Size in bytes	Values
describes logical (boolean) values	bool	1	true false

Note: *If you want to exclude negative values from the data span, indicate the keyword unsigned before the name of the type. For example, unsigned int. This type will include only positive values from 0 to 4,294,967,294.*

So, we found out what the types of data exist and which keywords in the C language are used to indicate them. In conclusion, we should note that C is case-sensitive (i.e. UP-PERCASE and lowercase letters are not the same things). Kindly note that all of the above described types of data are written in lowercase. Follow this, since int is a type of data and INT is an error.

We will discuss the application of data types in practice in the next topic.

7. Variables and Constants

We have already explored the types of data and now we know how to classify information for storage. We still need to determine how data is written in RAM and how they can be accessed in order to use or modify.

So, we agree on calling changing data VARIABLES, while we will call constant data CONSTANTS.

A ***variable*** is an area of the main memory, which has its own name and is used to store data that can be changed.

A ***constant*** is an area of the main memory, which has its own name and is used to store constant data.

Here is an example of constants: we all know the number of days of the week and months of the year ... It does not change under any circumstances, so these values are constants. However, our age is a variable. Today, anyone is 26 years old and he will be 27 in a year.

From the definitions it is clear that data are given names purposely to search them in the memory (by analogy with the fact that things in the baggage car are supplied with labels). In the programming environment, they are called identifiers. One of the first problems that parents of a newborn have to decide is to select a name for the baby. Whether a name leaves a mark on the character of the person and his destiny is a complex and controversial issue. You can hear absolutely opposite opinions on this subject. But the fact that a name (ID) to give a new variable (constant) should be intuitive and explain the assignment of the variable, will not be disputed by any more or less experienced programmer.

The names are given according to the strict rules. These rules must not be violated!

Rules for Names Composition

Only the following characters are allowed to use in the name:

1. Uppercase and lowercase letters of the Latin alphabet. Do not forget about the case-sensitivity of the language. For example, Age and age are two different names.
2. Digits. However, a digit cannot be used as the first character. That is, Name1 is acceptable, while 1Name is not acceptable.
3. The underline character _. The fact that you have to remember that space is also a character and this character is not valid in the name of the variable. The underline character will replace it in order to improve the expressiveness of names. For example, compare: ageofman and Age_Of_Man.

When determining a name for the variable, remember the following:

1. You cannot call a variable any keywords of the programming language. A keyword is a word that is reserved for the programming language syntax (int, float, double, and so on). In Visual Studio, the keywords are highlighted in blue, which will lead to confusion, at least.
2. Existence of two identifiers with the same names is undesirable.
3. You cannot use any other characters, except for allowable. (see above).

Declaring and Using Variables and Constants

Now we have all information to create (declare) a variable. The only things we need is to find out the general syntax:

1. ***data_type variable_name***; in this case, a memory cell of the size corresponding to the specified type will be dedicated in RAM. The name you have selected will be assigned to this cell. What will it contain? In the newly created variable, a random number determined by the operating system will be written. This number will be kept in the memory as long as you fill in the variable with another value, by means of a special assignment operator =.
2. ***datatype variable_name = value***; an opportunity to fill in the variable with value directly when created is available as well. We call this process initialization.
3. ***const data_type variable_name = value***; this is a declaration of the constant. The main point is that the keyword const is specified before, regardless of the type of data. Besides, the constant must be initialized during the creation. Its value cannot be changed later.

Displaying Value of Variable

Displaying value of variable is performed by using cout<<

```
cout<<variable_name; // quotes are not indicated
```

You can display the contents of several variables through <<

```
cout<<variable_name1<<variable_name2;
// quotes are not indicated
```

You can alternate the display of the contents of the variables with text messages and Escape sequences through <<

```
cout<<"Text"<<variable_name1
<<"Text"<<variable_name2<<"\n";
```

Displaying the contents of the constants is carried out on a complete analogy with the variables.

Practical Examples

Here are some examples of creation and initialization of variables and constants for different data types.

Integer Variables and Constants

We meet integers everywhere: age, number of seats, number of rooms, number of days of the week, etc.

The variables that will store integers are DECLARED as follows:

- ***int Age;***

What does this line mean? It means that a variable named Age will store an integer value. The word int declares the TYPE of the value of the variable called Age.

Now, for example, we want to add the value 34 to the variable Age. How can we do it?

- ***Age = 34;***

This line is read as follows: "assign a value of 34 to the variable Age". Once again, let us look at the assignment statement: Age = 34. To the left of the equal sign, it is the name of the variable, to which the value is assigned. To the right of the sign, it is the value that is assigned.

The constant that will store the integer is declared as follows:

- ***const int Count_Days_in_Week=7;***

What does this line means? The word const stresses that constant is declared. int says that the constant is an integer. This is followed by the constant name Count_Days_in_Week and its value of 7.

We now consider how to calculate the value of the variable. What is it for? A simple example: how to calculate a number of hours in the year 2000? Do you want to count the number of ourselves?

In fact, it is quite easy to get the computer to do it. We are required to write only the formula of this calculation.

In 2000, there are 366 days, with 24 hours per day. The formula for calculating the number of hours in 2000 will be as follows: multiply 366 by 24.

In C, * is used as a multiplication sign (asterisk, the combination Shift + 8).

We will develop a program that counts how many hours are in 2000.

Before you create the program, it is recommended to briefly draft its algorithm.

Algorithm is a sequence of actions aimed at the solution of the current problem.

Given: the number of days in the year is 366. This value will not change, so we declare it an integer constant named DayIn_2000Year. The number of hours in the day and night is 24. It is not changed as well. We declare it an integer constant named HourInDay. Our program will contain the only variable and we will use it to write the result of the calculation in. We will call this variable HourIn_Year2000. It will be an integer (int).

The algorithm is as follows:

1. Declaring and initializing variables and constants.
2. Calculation of results.
3. Displaying the results.

You can think of the names for the variables by yourself (just do not forget about the rules of the composition of the names for variables).

Now, as always, we will create a new project and enter the following code:

```
// Title
#include  <iostream>
// definition of the namespace, which contains cout<<
using namespace std;
// Main function
void main()
{
     // Print a blank line
     cout<<"\n";
     // Declare integer constants
     int DayIn_2000Year=366;
     int HourInDay=24;
     // declare an integer variable
     int  HourIn_Year2000;
     // Calculate the desired value and
     // Put it in the variable HourIn_Year2000
     HourIn_Year2000=DayIn_2000Year*HourInDay;
     // Display the variable  HourIn_Year2000
     cout<<"\t\t In 2000 year "<< HourIn_Year2000;
     cout<<" hours\n ";
}
```

That is all! Compile the program!

Real Variables and Constants

Example of declaring and initializing

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

weight_atom= 2.5194e+017;

What is the number of 2.5194e + 017?

This is a note of real numbers. It is called exponent form of numbers notation. We will tell you the secret of decryption of the writing. This character set describes the number 251940000000000000 or 2,1594 × 10^{17}. 3.4E–008 stands as: 3,4 × 10^{-8}, which is similar to 3.4:10^{8}.

–1.5E + 003 stands as –1,5 × 10^{3}.

Floating-point numbers of the type float can vary from -3,4×10^{38} до 3,4×10^{38}.

The values from -3,4 × 10^{-38} to 3,4 × 10^{-38} are considered to be zero.

Now let us work with real numbers in practice:

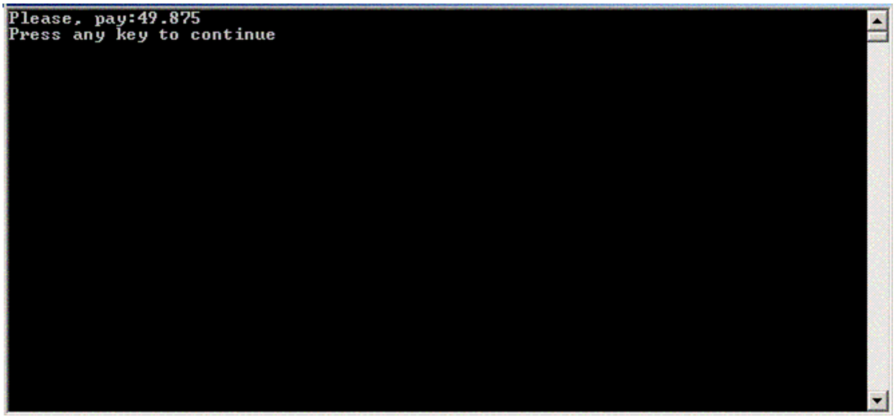
Write a program that will calculate the purchase price. Let the program use the price of the item (Cost), the amount of purchased goods (Count), and, given the discount (Discount), calculates the purchase price (Price).

Create a new project ***Purchase*** and enter the text of the next program.

```
// Title
#include <iostream>
// definition of the namespace, which contains cout <<
```

```
using namespace std;
// Main function
void main ()
{
    // Declare the variable Discount
    float Discount=0.05;
    // Declare the variable Cost
    float Cost=10.50;
    // Declar the variable Count
    int Count=5;
    // Declare the variable Price
    float Price;
    // Compute the value of the variable Price
    Price=Count*Cost-Count*Cost*Discount;
    // Display the final cost of the item at a discount
    cout<<"Please,pay:"<<Price<<"\n";
}
```

Compile the program and run it for execution. What you should see on the screen is shown below.



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

Character and Logical Variables and Constants

In this lesson, we will not give examples of the use of symbolic and logical variables and constants. Their purpose will be described in greater detail in the future. We will discuss only declaration and initialization.

```
// logical variable
bool Flag;
Flag=true;
// One character is always specified in single quotes
char Symbol='A';
/* Escape sequence is considered by the compiler as
   a single character and can be accordingly recorded
   in the variable or constant of the type char * /
const char NewLine='\n';
cout<<NewLine // displays an empty line
```

8. Data Input

You are already familiar with the operation of displaying information on the computer screen — cout, but most programs require not only to display information on the screen, but also be able to enter data into the computer using keyboard. The previous section shows the program for calculating the discount. Naturally, it would be convenient to enter parameters such as price and quantity of the goods from the keyboard on the stage of the program execution. Let us look how you can do it.

If we need to enter data into the computer, we will use a command cin. How to use it? The syntax of the input statement is as follows:

```
cin>>var_name;
```

var_name points to the variable, into which the data entered from the keyboard must be put:

For example:

```
cin>>Age;
```

This command puts the number entered from the keyboard into the variable named Age. To enter a number into the variable Number, just print the following command:

```
cin>>Number;
```

Entering several variables is recorded as follows:

```
cin>>var_name 1>>var_name 2>>... >>var_nameN;
```

The list with the variable names must contain the names of all variables, into which you want to enter data from the keyboard. The list of names can consist of any number of variable names separated by a combination of characters >>.

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

Let us add data keyboard input to the program ***Purchase***:

```
// Title
#include <iostream>
// definition of namespace, which contains cout <<
using namespace std;
// Main function
void main ()
{
    //Declare the variable Discount
    float Discount=0.05;
    //Declare the variable Cost
    float Cost=10.50;
    //Prompt to enter the price of the item
    cout<<"What's the cost?\n";
    //Enter the value in the variable Cost
    cin>>Cost;
    //Declare the variable Count
    int Count=5;
    //Prompt to enter the amount
    cout<<"How much?\n";
    //Enter the value in the variable Count
    cin>>Count;
    //Declare the variable Price
    float Price;
    //Compute the value of the variable Price
    Price=Count*Cost-Count*Cost*Discount;
    //Display the final cost of the item at a discount
    cout<<"Please, pay:"<<Price<<"\n";
}
```

Now you see the feature of the operation of the statement cin>>. Once the program meets this statement, it stops and waits for user response until the user enters data and clicks "Enter". Only then execution is continued.

Let us again have some practice with input and output by the example. Let us write a program-cheater: the program offers to play pick-a-number game: those, who think of greater number, will win.

Create a new project ***Game*** and type the following text:

```
// Title
#include <iostream>
// definition of namespace, which contais cout <<
using namespace std;
// Main function
void main ()
{
    //Invitation "Let's play!"
    cout<<"Let's play!\n";
    //Declare the variable i
    int i;
    //Prompt "Enter the number"
    cout<<"Enter a number:";
    //Enter the number
    cin >> i;
    //Display the number that the computer proposes
    cout<<"I have "<<i+1<<"\n";
    //Display the result of the game
    cout<<"I'm winner!\n";
}
```

Compile the program. Our program is easy to handle: enter any number and it will constantly turn out that the computer offers greater number and wins. Here is what you will see on the screen, when running the program, if you enter the number 67 to the request "Enter a number:":

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

Why does the computer always win?
Let us look at the line

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

It shows the value of the variable i, which value that you entered from the keyboard, increased by 1, that is, the computer always displays the number greater by 1 than the number that you have entered from the keyboard.

If you replace the expression i + 1 by the expression i-1, you will always win, since the number displayed by the computer will always be one less than the number that you would enter from the keyboard.

In conclusion, we want to draw your attention to the statements + (plus) and – (minus). They are used for addition and subtraction. In C, there is the division operator /. This information will help you with your home assessment, and we will talk about the statement in more detail in the following lessons..

9. Literals

Literals are fixed values that the program is unable to change. For each type of the C language, there are literals, including character and Boolean types, integers and floating point numbers. It may seem paradoxically, but there is no data type to store strings in C, but there are string literals.

Some examples:

5	integer literal — int
5l	l or L means long
true	Boolean literal — bool
5.0	floating-point literal, taken as double
5.0f	f or F — with a floating point, taken as float
0.3e-2	floating point literal double, e or E separates exponential part
'd'	character literal
"Visual"	string literal — a quoted set of random characters. The compiler takes it as a set of characters and does not execute it, even the quotes contain some keywords or operations.

Examples of the code that contains literals.

```
// "abrakadabra" - a string literal '\n' -
// a character literal
cout<<"abrakadabra"<<'\n';
int a = 2; // 2 - int literal
```

The presentation of the material of the first lesson is completed now. We strongly recommend you to accomplish your home assessment, provided in the next section. Good luck!

10. Home Assessment

1. Write a program that computes the average of the two numbers.
2. Write a program that translates hryvnias to American dollars, euros, Russian rubles.
3. Display the following text:

   ```
   "To be or not to be"
    \Shakespeare\
   ```

4. In C, there is no squaring operation. Write a program that computes the square of any entered number.
5. Enter three numbers and display the sum and the product of these numbers.

Lesson No. 1

Introduction to Programming