C++

به زبان ساده

فهرست مطالب

برای دریافت فایل‌ها و آپدیت‌های جدید این کتاب به سایت **www.w3-farsi.com** مراجعه فرمایید.

راه‌های ارتباط با نویسنده

وب سایت: www.w3-farsi.com

لینک تلگرام: https://telegram.me/ebrahimi_younes

ID تلگرام: @ebrahimi_younes

پست الکترونیکی: younes.ebrahimi.1391@gmail.com

تقدیم به:

همسر و پسر عزیزم

مبانی زبان سی پلاس پلاس

C++ چیست

C++ یک زبان برنامه نویسی شیءگراست که در سال ۱۹۸۵ توسط Bjarne Stroustup دانشمند دانمارکی به وجود آمد. C++ نسخه توسعه یافته زبان C می‌باشد و بیشتر کدهای زبان C به راحتی می‌تواند در C++ کامپایل شود. در C++ از ویژگی‌های مهمی که به C اضافه شده است می‌توان به برنامه نویسی شیءگرا، سربارگذاری عملگرها، وراثت چندگانه و مدیریت خطاها اشاره نمود. توسعه C++ در سال ۱۹۷۹ آغاز شد و ۷ سال پس از زبان C به نمایش گذاشته شد. با وجود قدیمی بودن زبان‌های C و C++، هنوز هم به صورت گسترده‌ای در نرم افزارهای صنعتی مورد استفاده قرار می‌گیرد. این زبان‌ها برای ساخت هر چیزی از سیستم عامل گرفته تا نرم افزارهای توکار، برنامه‌های دسکتاپ و بازی‌ها مورد استفاده قرار می‌گیرد.

در مقایسه با زبان‌های جدیدتر، برنامه‌های نوشته شده با C++ اغلب پیچیده‌تر می‌باشند و زمان بیشتری برای توسعه نیاز دارد. در عوض، C++ زبانی است که به شما اجازه می‌دهد که هم به صورت High-level (نزدیک به زبان انسان) و هم به صورت low-level (نزدیک به زبان ماشین) سخت افزار را تحت کنترل خود قرار دهید. همچنین با پشتیبانی از سبک‌های مختلف برنامه نویسی از جمله رویه‌ای، شیءگرا یا عمومی، دست برنامه نویس را در انتخاب سبک مورد نظرش آزاد می‌گذارد. اکنون ۵ نسخه از استاندارد این زبان منتشر شده است؛ و استاندارد C++17 نیز برای انتشار در سال ۲۰۱۷ برنامه ریزی شده است.

نام غیر رسمی	استاندارد++C	سال
C++98	ISO/IEC 14882:1998	1998
C++03	ISO/IEC 14882:2003	2003
C++07/TR1	ISO/IEC TR 19768:2007	2007
C++11	ISO/IEC 14882:2011	2011
C++14	ISO/IEC 14882:2014	2014
C++17	هنوز تعیین نشده.	2017

برای اجرای کدهای C++ نیاز به یک کامپایلر داریم. کامپایلرها و محیط‌های برنامه نویسی (IDE) گوناگونی برای زبان C++ وجود دارند از بین معروف‌ترین آن‌ها می‌توان موارد زیر اشاره نمود:

- Turbo C
- Turbo C++
- Borland C++
- Microsoft visual Studio

زبان C++ وابسته به یک سیستم عامل نیست یعنی شما بعد از نوشتن برنامه خود به زبان++C ، اگر کد استانداری نوشته باشید می‌توانید با توجه به سیستم عامل، کدتان را کامپایل کنید. می‌توان کد C++ را در هر محیطی، مثلاً NotePadدر ویندوز و یا gEdit در گنو/لینوکس نوشته و بعد آن را بوسیله یک کامپایلر کامپایل کنیم، ولی برای راحتی کار ما می‌توانیم از یک IDE مناسب، نیز بهره ببریم. البته در این سری آموزشی ما از بهترین IDE برای کامپایل کدها استفاده می‌کنیم.

ویژوال استودیو

ویژوال استودیو محیط توسعه یکپارچه‌ای است، که دارای ابزارهایی برای کمک به شما برای توسعه برنامه‌های C++ می‌باشد. شما می‌توانید یک برنامه C++ را با استفاده از برنامه notepad یا هر برنامه ویرایشگر متن دیگر بنویسید و با استفاده از کامپایلر C++ از آن استفاده کنید، اما این کار بسیار سخت است چون اگر برنامه شما دارای خطا باشد خطایابی آن سخت می‌شود. توصیه می‌کنیم که از محیط ویژوال استودیو برای ساخت برنامه استفاده کنید چون این محیط دارای ویژگی‌های زیادی برای کمک به شما جهت توسعه برنامه‌های C++ می‌باشد. تعداد زیادی از پردازش‌ها که وقت شما را هدر می‌دهند به صورت خودکار توسط ویژوال استودیو انجام می‌شوند.

یکی از این ویژگی‌ها اینتلی سنس (Intellisense) است که شما را در تایپ سریع کدهایتان کمک می‌کند. ویژوال استودیو برنامه شما را خطایابی می‌کند و حتی خطاهای کوچک (مانند بزرگ یا کوچک نوشتن حروف) را برطرف می‌کند، همچنین دارای ابزارهای طراحی برای ساخت یک رابط گرافیکی است که بدون ویژوال استودیو برای ساخت همچنین رابط گرافیکی باید کدهای زیادی نوشت. با این برنامه‌های قدرتمند بازدهی شما افزایش می‌یابد و در وقت شما با وجود این ویژگیهای شگفت انگیز صرفه‌جویی می‌شود.

در حال حاضر آخرین نسخه ویژوال استودیو Visual Studio 2017 است. این نسخه به دو نسخه Visual Studio Professional (ارزان قیمت) و Visual Studio Enterprise (گرانقیمت) تقسیم می‌شود و دارای ویژگی‌های متفاوتی هستند. خبر خوب برای توسعه‌دهندگان نرم‌افزار این است که مایکروسافت تصمیم دارد که ویژوال استودیو را به صورت متن باز ارائه دهد. یکی از نسخه‌های ویژوال استودیو، Visual Studio Community می‌باشد که آزاد است و می‌توان آن را دانلود و از آن استفاده کرد. این برنامه ویژگی‌های کافی را برای شروع برنامه‌نویسی C++ در اختیار شما قرار می‌دهد. این نسخه (Community) کامل نیست و خلاصه‌شده نسخه اصلی است. به هر حال استفاده از Visual Studio Community که جایگزین Visual Studio Express شده و به نوعی همان نسخه Visual Studio Professional است، برای انجام تمرینات این سایت کافی است.

Visual Studio Enterprise 2017 دارای محیطی کامل‌تر و ابزارهای بیشتری جهت عیب‌یابی و رسم نمودارهای مختلف است که در Visual Studio Community وجود ندارند. ویژوال استودیو فقط به C++ خلاصه نمی‌شود و دارای زبان‌های برنامه‌نویسی دیگری از جمله ویژوال بیسیک نیز می‌باشد.

دانلود و نصب ویژوال استودیو

در این درس می‌خواهیم نحوه دانلود و نصب نرم افزار Visual Studio Community 2017 را آموزش دهیم. در جدول زیر لیست نرم افزارها و سخت افزارهای لازم جهت نصب ویژوال استودیو ۲۰۱۷ آمده است:

سیستم عامل	سخت افزار
Windows 10	1.6 GHz or faster processor
Windows 8.1	1 GB of RAM (1.5 GB if running on a virtual machine)
Windows 8	4 GB of available hard disk space
Windows 7 Service Pack 1	5400 RPM hard disk drive

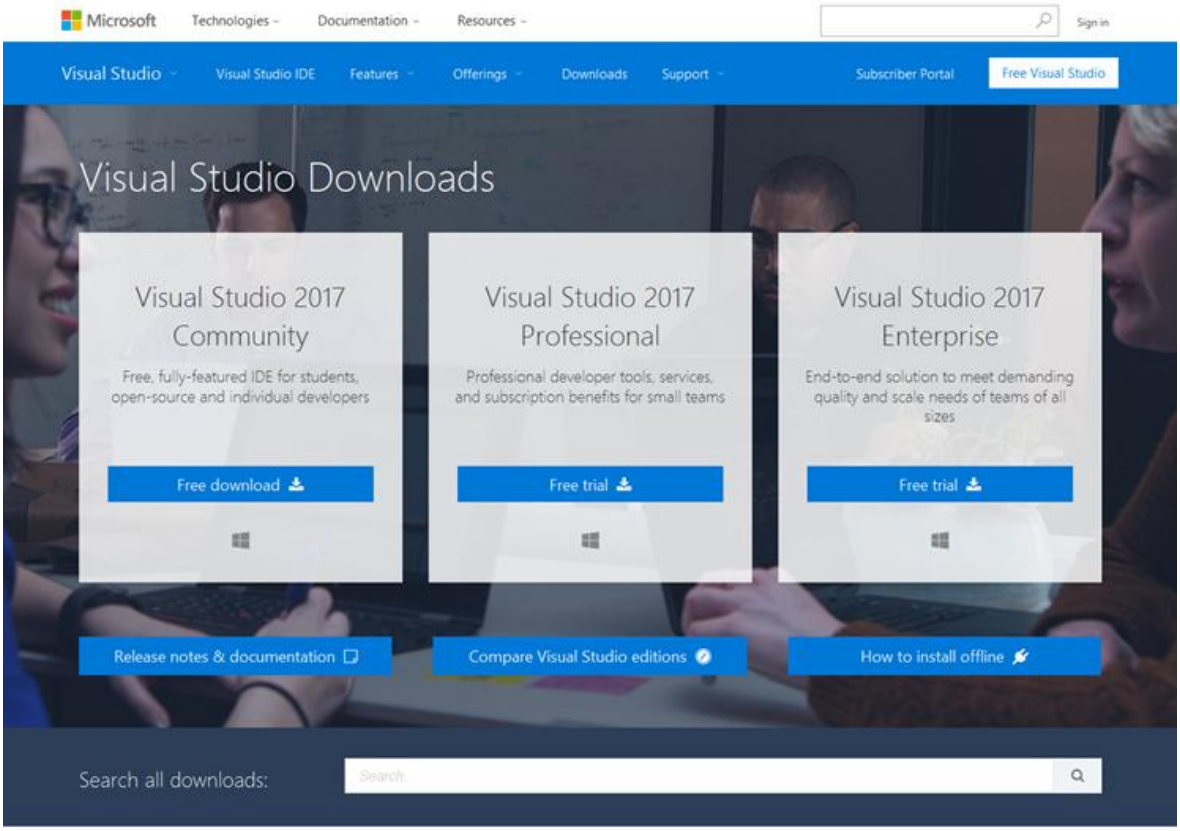
Windows Server 2012 R2	DirectX 9-capable video card that runs at 1024 x 768 or higher display resolution
Windows Server 2012	
Windows Server 2008 R2 SP1	

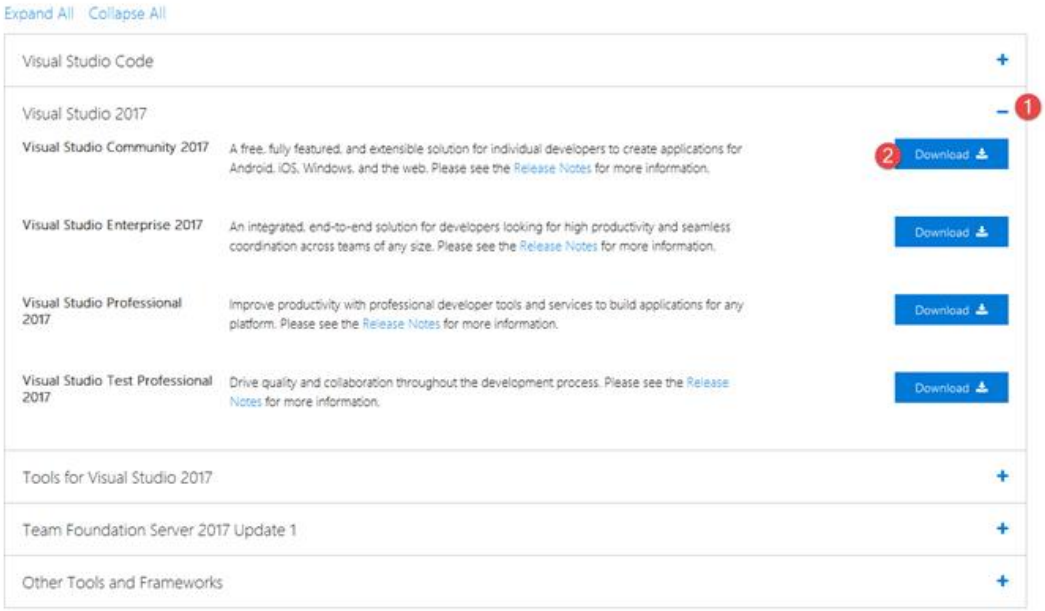
دانلودVisual Studio Community 2017

Visual Studio Community 2017 به صورت آزاد در دسترس است و می‌توانید آن را از لینک زیر دانلود کنید:

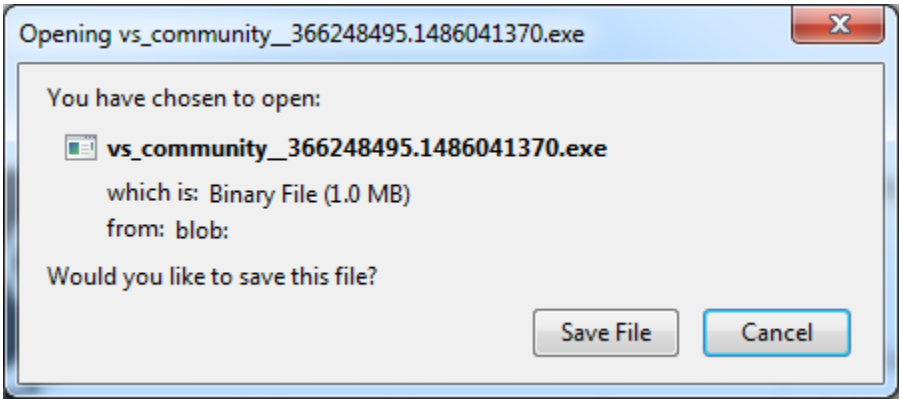
https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx

با کلیک بر روی لینک بالا صفحه ای به صورت زیر ظاهر می‌شود که در داخل این صفحه می‌توان با کلیک بر روی Visual Studio Community 2017 آن را دانلود کرد:

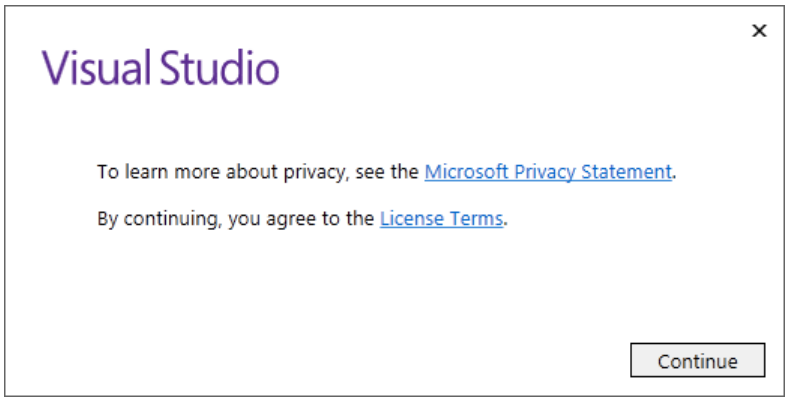


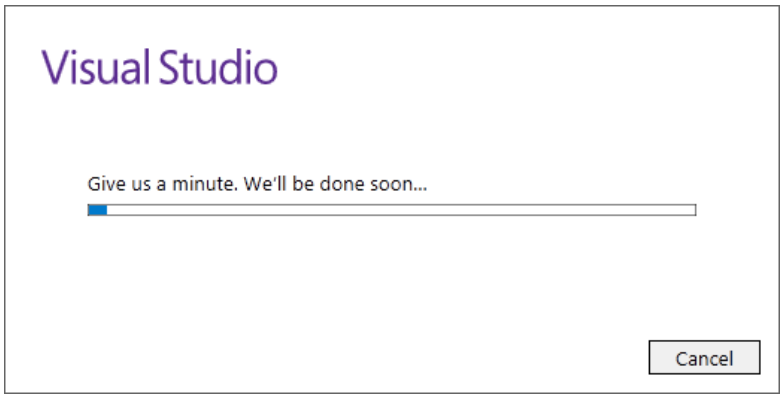


بعد از کلیک بر روی گزینه Download یک صفحه به صورت زیر باز می‌شود و از شما می‌خواهد که فایلی با نام vs_community.exe را ذخیره کنید:

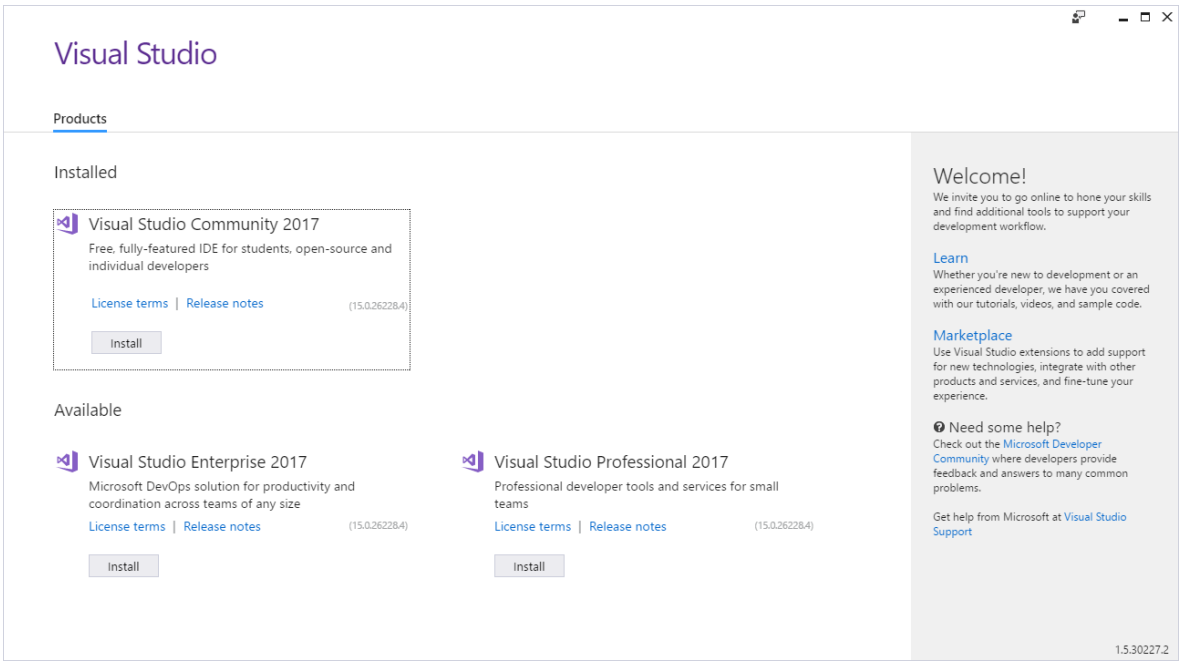


با ذخیره و اجرای این فایل مراحل نصب Visual Studio Community 2017 آغاز می‌شود (Visual Studio Community 2017 حدود ۵ گیگابایت حجم دارد و برای دانلود آن به یک اینترنت پر سرعت دارید):

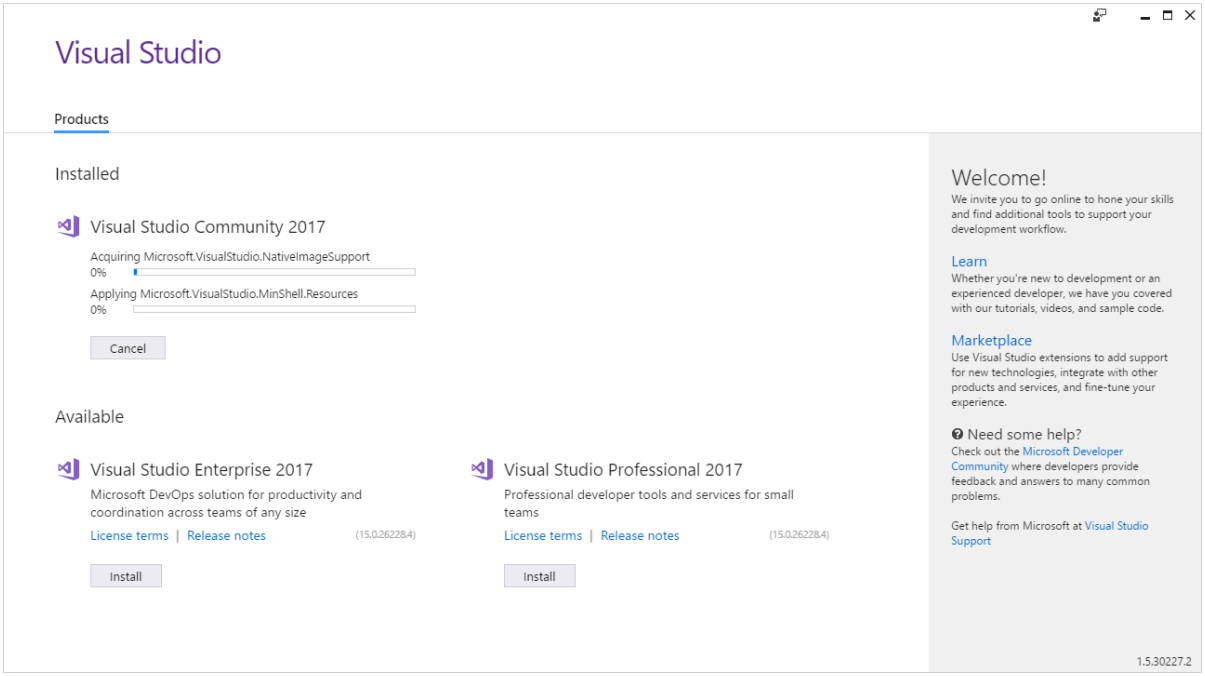




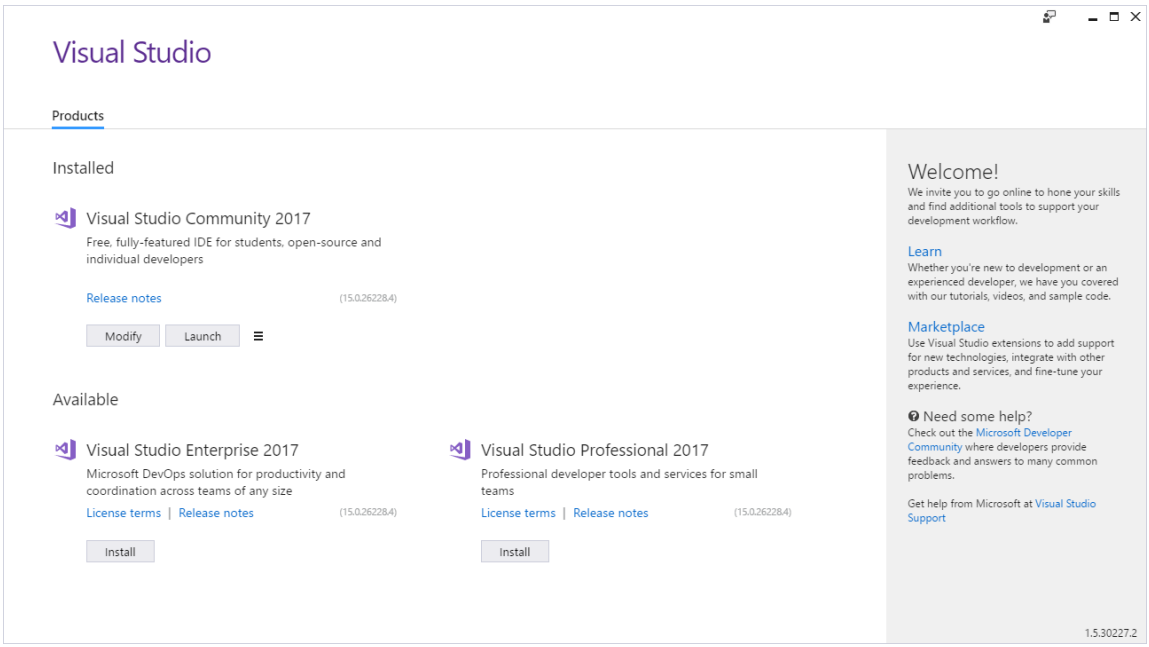
بعد از گذراندن دو صفحه بالا صفحه ای به صورت زیر باز می‌شود که در آن نسخه‌های مختلف ویژوال استودیو به شما نمایش داده می‌شود. بر روی گزینه Install روبروی Visual Studio Community کلیک کنید:



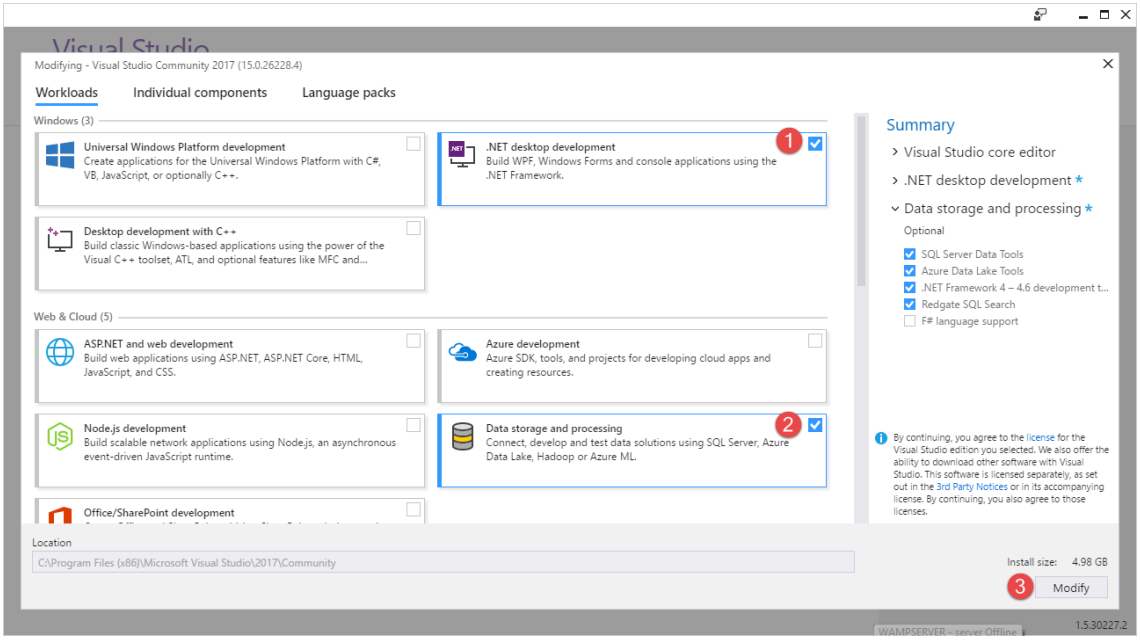
بعد از کلیک بر روی دکمه `Install` مرحله نصب شروع می‌شود:



بعد از اتمام مرحله بالا صفحه ای به صورت زیر باز می‌شود:



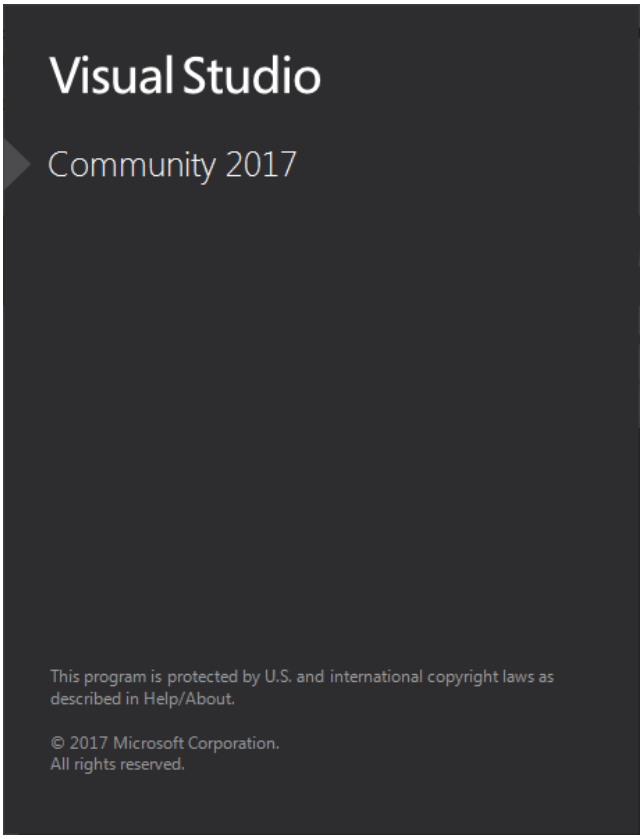
در صفحه بالا بر روی گزینه `Modify` کلیک کنید و گزینه‌های زیر را تیک بزنید و سپس بر روی دکمه `Modify` کلیک کنید:



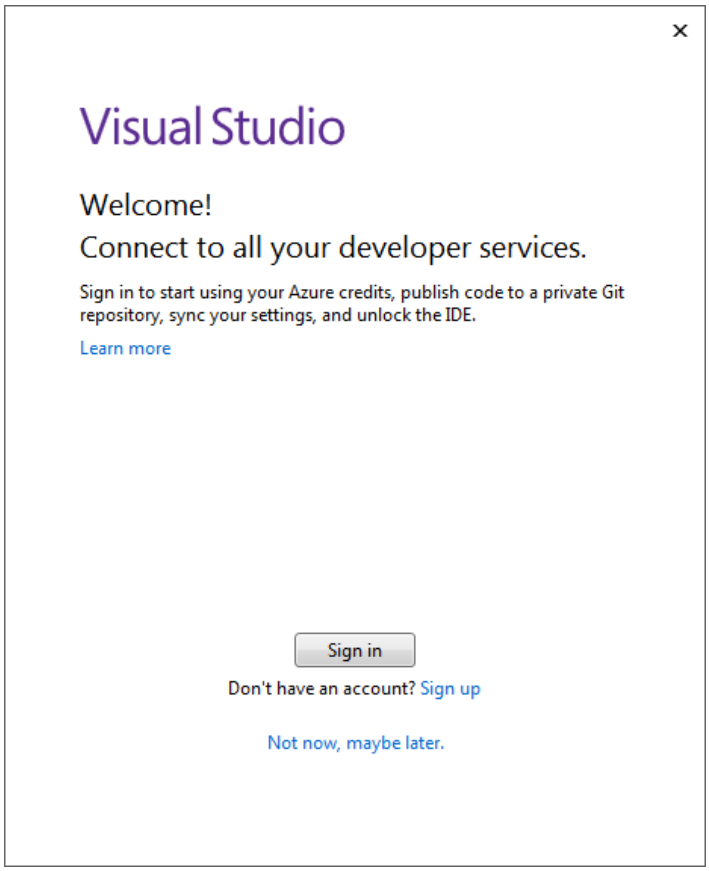
بعد از این مرحله ویژوال استودیو به صورت کامل نصب شده و شما می‌توانید از آن استفاده کنید.

شروع کار با Visual Studio Community

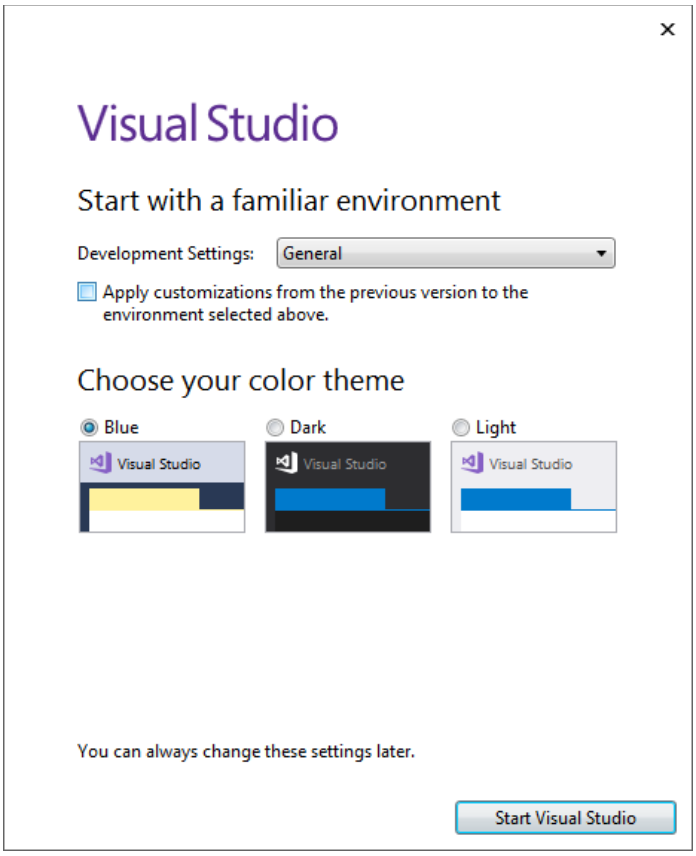
برنامه ویژوال استودیو را اجرا کرده و منتظر بمانید تا صفحه آن بارگذاری شود:



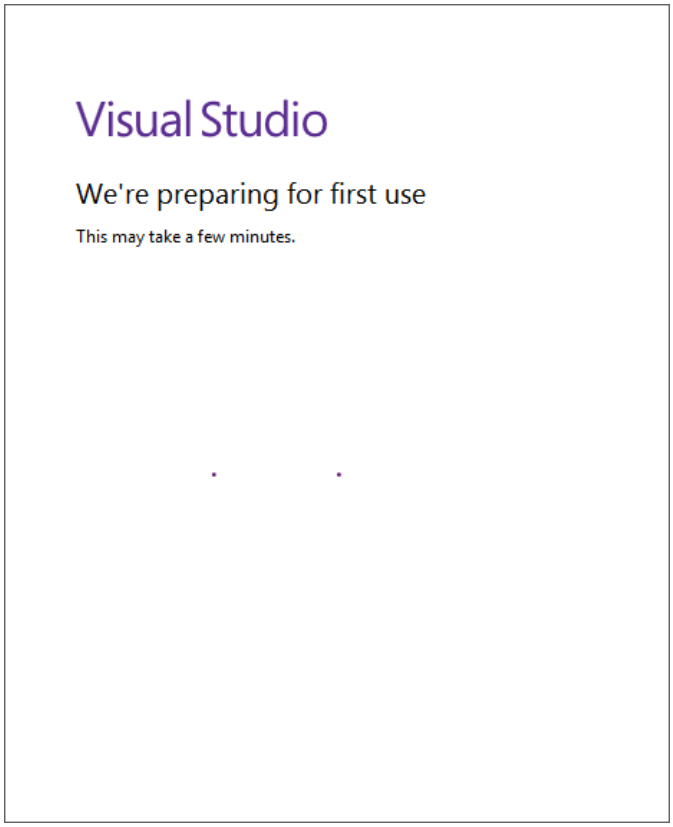
اگر دارای یک اکانت مایکروسافت باشید می‌توانید تغییراتی که در ویژوال استودیو می‌دهید را در فضای ابری ذخیره کرده و اگر آن را در کامپیوتر دیگر نصب کنید، می‌توانید با وارد شده به اکانت خود، تغییرات را به صورت خودکار بر روی ویژوال استودیویی که تازه نصب شده اعمال کنید. البته می‌توانید این مرحله را با زدن دکمه `Not now, maybe later` رد کنید:



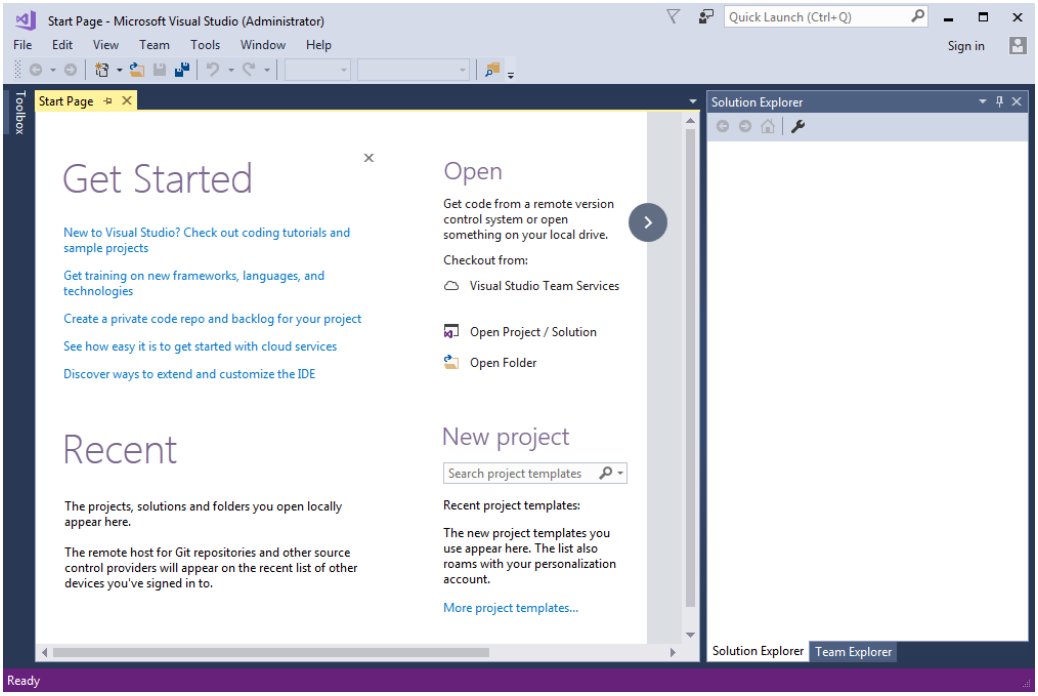
شما می‌توانید از بین سه ظاهر از پیش تعریف شده در ویژوال استودیو یکی را انتخاب کنید. من به صورت پیشفرض ظاهر `Blue` را انتخاب می‌کنم ولی شما می‌توانید بسته به سلیقه خود، ظاهر دیگر را انتخاب کنید:



بعد از زدن دکمه `Start Visual Studio` صفحه ای به صورت زیر ظاهر می‌شود:

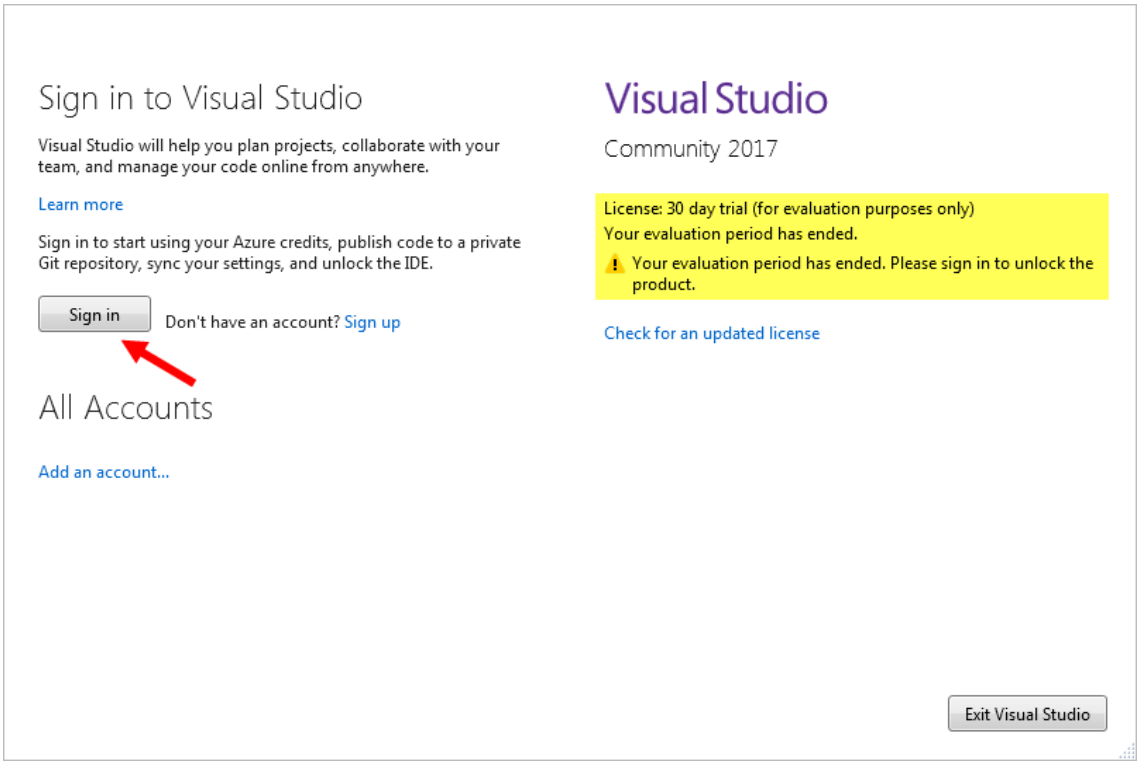


بعد از بارگذاری کامل `Visual Studio Community` صفحه اصلی برنامه به صورت زیر نمایش داده می‌شود که نشان از نصب کامل آن دارد:



قانونی کردن ویژوال استودیو

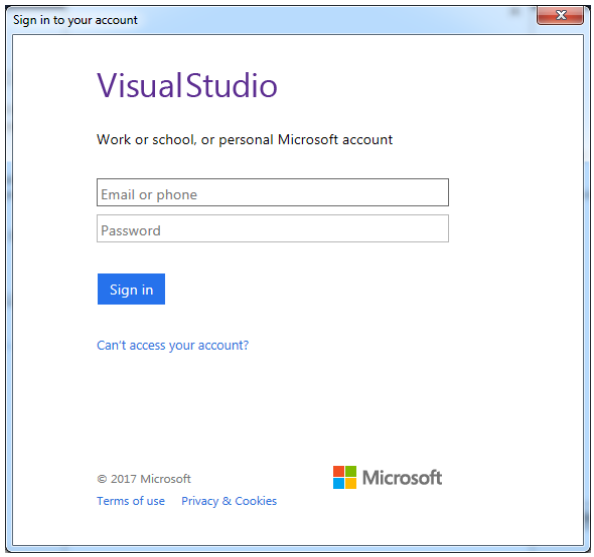
Visual Studio Community 2017 رایگان است. ولی گاهی اوقات ممکن است با پیغامی به صورت زیر مبنی بر منقضی شدن آن مواجه شوید:



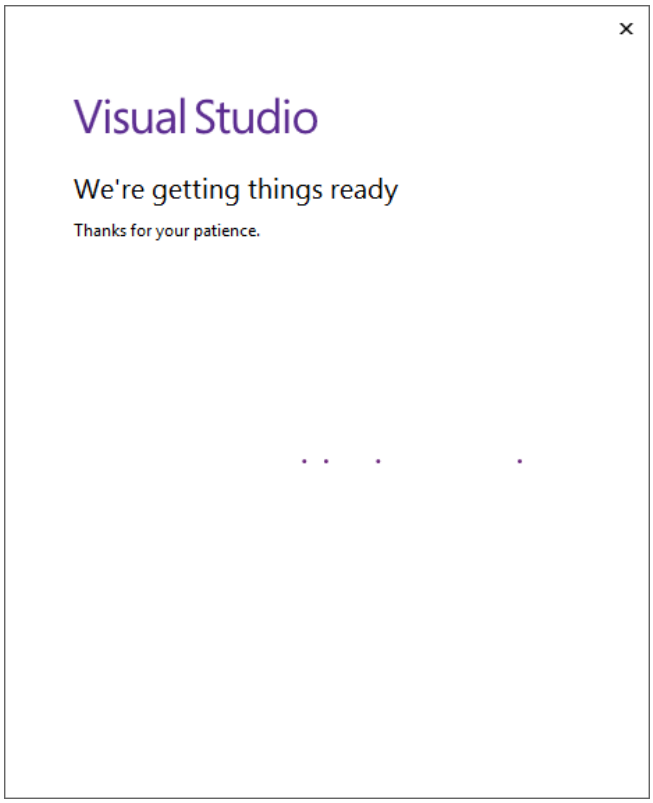
همانطور که در شکل بالا مشاهده می‌کنید، بر روی دکمه Signin کلیک می‌کنید تا وارد اکانت مایکروسافت خود شوید. اگر اکانت ندارید، می‌توانید از لینک زیر یک اکانت ایجاد کنید:

```
http://www.w3-farsi.com/?p=22201
```

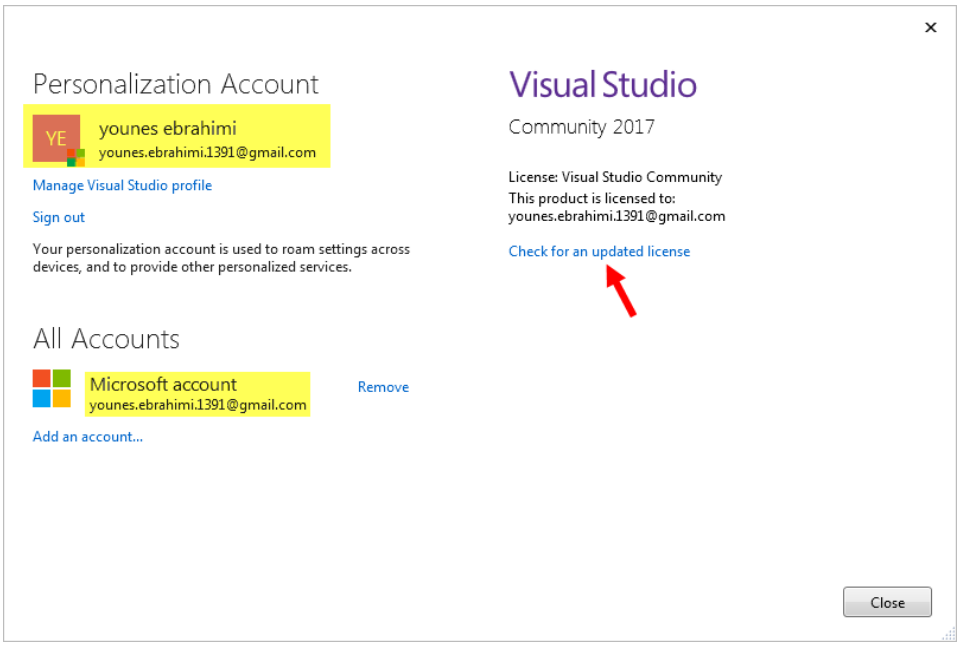
بعد از ایجاد اکانت همانطور که در شکل بالا مشاهده می‌کنید، بر روی گزینه Singin کلیک می‌کنیم. با کلیک بر روی این گزینه صفحه ای به صورت زیر ظاهر می‌شود که از شما مشخصات اکانتتان را می‌خواهد، آن‌ها را وارد کرده و بر روی گزینه Singin کلیک کنید:



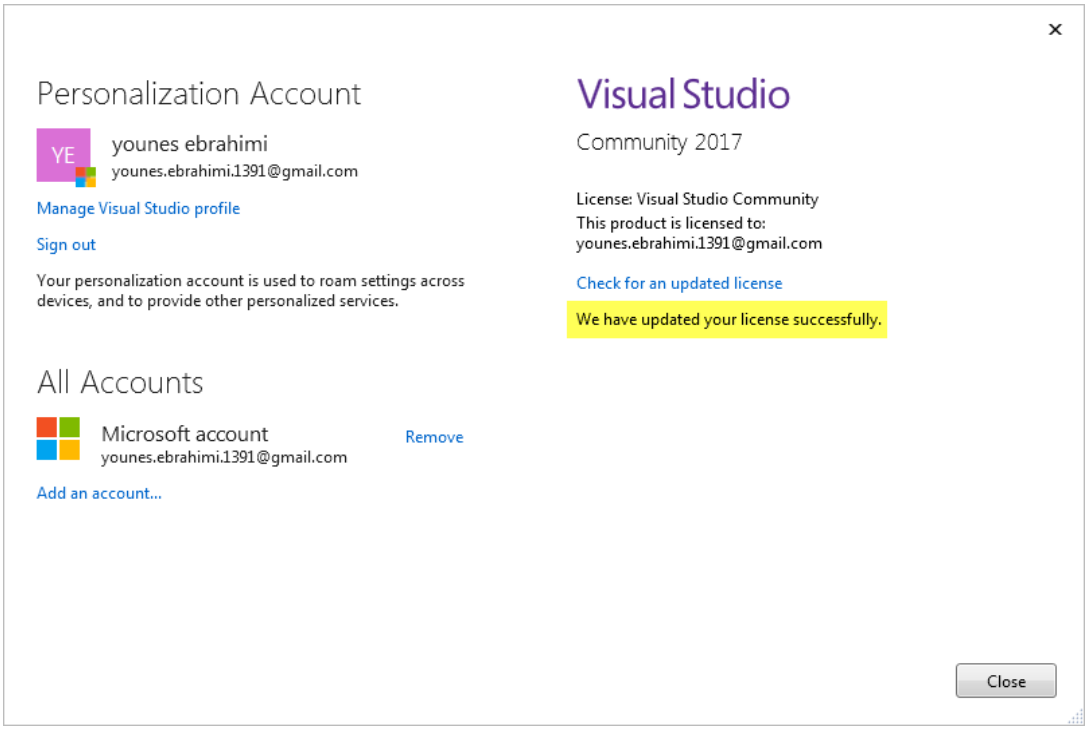
با کلیک بر روی گزینه `Signin` پنجره ای به صورت زیر نمایش داده می‌شود، منتظر می‌مانید تا پنجره بسته شود:



با بسته شدن پنجره بالا، پنجره ای به صورت زیر ظاهر می‌شود که مشخصات اکانت شما در آن نمایش داده می‌شود، که نشان از ورود موفقیت آمیز شما دارد. در این صفحه بر روی گزینه `Check an updated license` کلیک کنید:



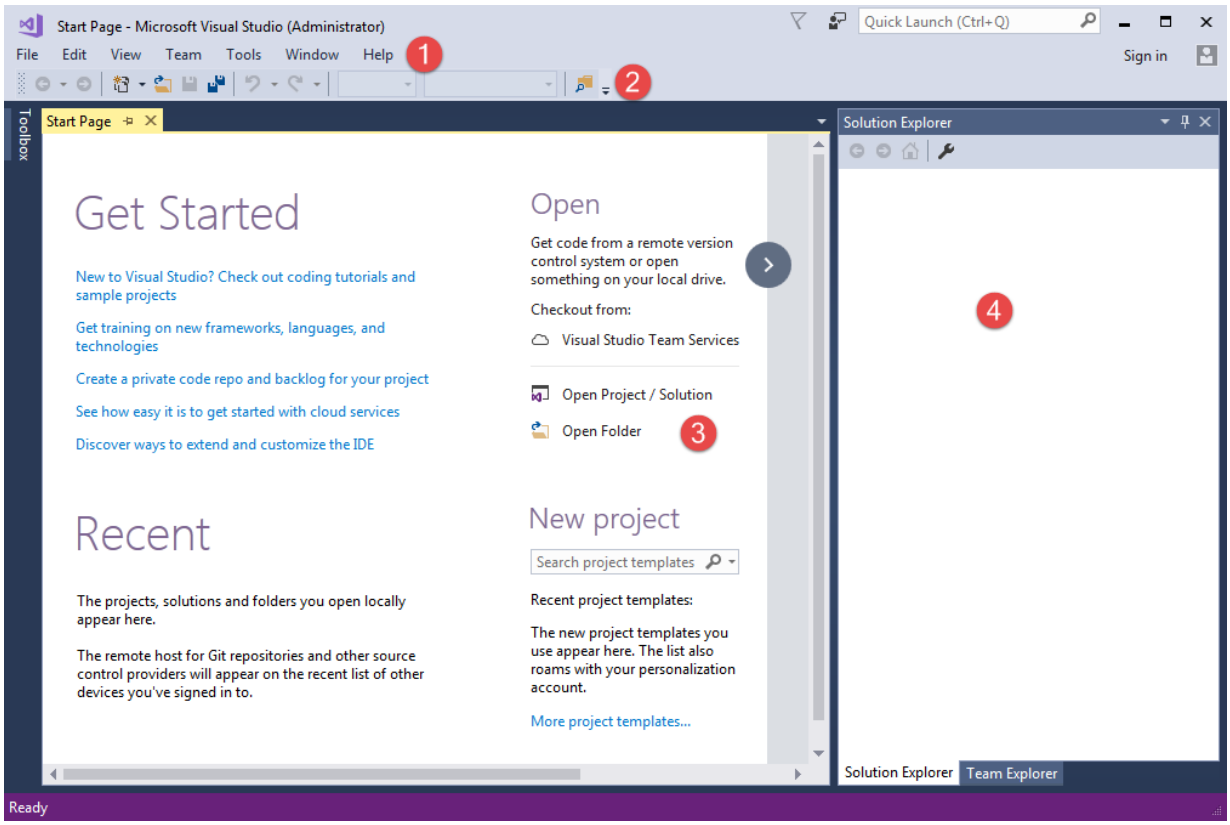
با کلیک بر روی این گزینه بعد از چند ثانیه پیغام `we have updated your license successfully` نمایش داده می‌شود و به این صورت ویژوال استودیو قانونی می‌شود:



به ویژوال استودیو خوش آمدید

در این بخش می‌خواهیم درباره قسمت‌های مختلف محیط ویژوال استودیو به شما مطالبی آموزش دهیم. لازم است که با انواع ابزارها و ویژگیهای این محیط آشنا شوید. برنامه ویژوال استودیو را اجرا کنید:

بعد از اینکه صفحه بالا بسته شد وارد صفحه آغازین ویژوال استودیو می‌شویم:



این صفحه بر طبق عناوین خاصی طبقه بندی شده که در مورد آن‌ها توضیح خواهیم داد.

منو بار(Menu Bar)

(1) Menu Bar که شامل منوهای مختلفی برای ساخت، توسعه، نگهداری، خطایابی و اجرای برنامه‌ها است. با کلیک بر روی هر منو دیگر منوهای وابسته به آن ظاهر می‌شوند. به این نکته توجه کنید که منوبار دارای آیتم‌های مختلفی است که فقط در شرایط خاصی ظاهر می‌شوند. به عنوان مثال آیتم‌های منوی Project در صورتی نشان داده خواهند شد که پروژه فعال باشد. در زیر برخی از ویژگیهای منوها آمده است:

منو	توضیح
File	شامل دستوراتی برای ساخت پروژه یا فایل، باز کردن و ذخیره پروژه‌ها و خروج از آن‌ها می‌باشد
Edit	شامل دستوراتی جهت ویرایش از قبیل کپی کردن، جایگزینی و پیدا کردن یک مورد خاص می‌باشد

View	به شما اجازه می‌دهد تا پنجره‌های بیشتری باز کرده و یا به آیتم‌های toolbar آیتمی اضافه کنید .
Project	شامل دستوراتی در مورد پروژه ای است که شما بر روی آن کار می‌کنید .
Debug	به شما اجازه کامپایل، اشکال زدایی و اجرای برنامه را می‌دهد
Data	شامل دستوراتی برای اتصال به دیتابیس‌ها می‌باشد .
Format	شامل دستوراتی جهت مرتب کردن اجزای گرافیکی در محیط گرافیکی برنامه می‌باشد .
Tools	شامل ابزارهای مختلف، تنظیمات و ...برای ویژوال استودیو می‌باشد .
Window	به شما اجازه تنظیمات ظاهری پنجره‌ها را می‌دهد .
Help	شامل اطلاعاتی در مورد برنامه ویژوال استودیو می‌باشد

The Toolbars

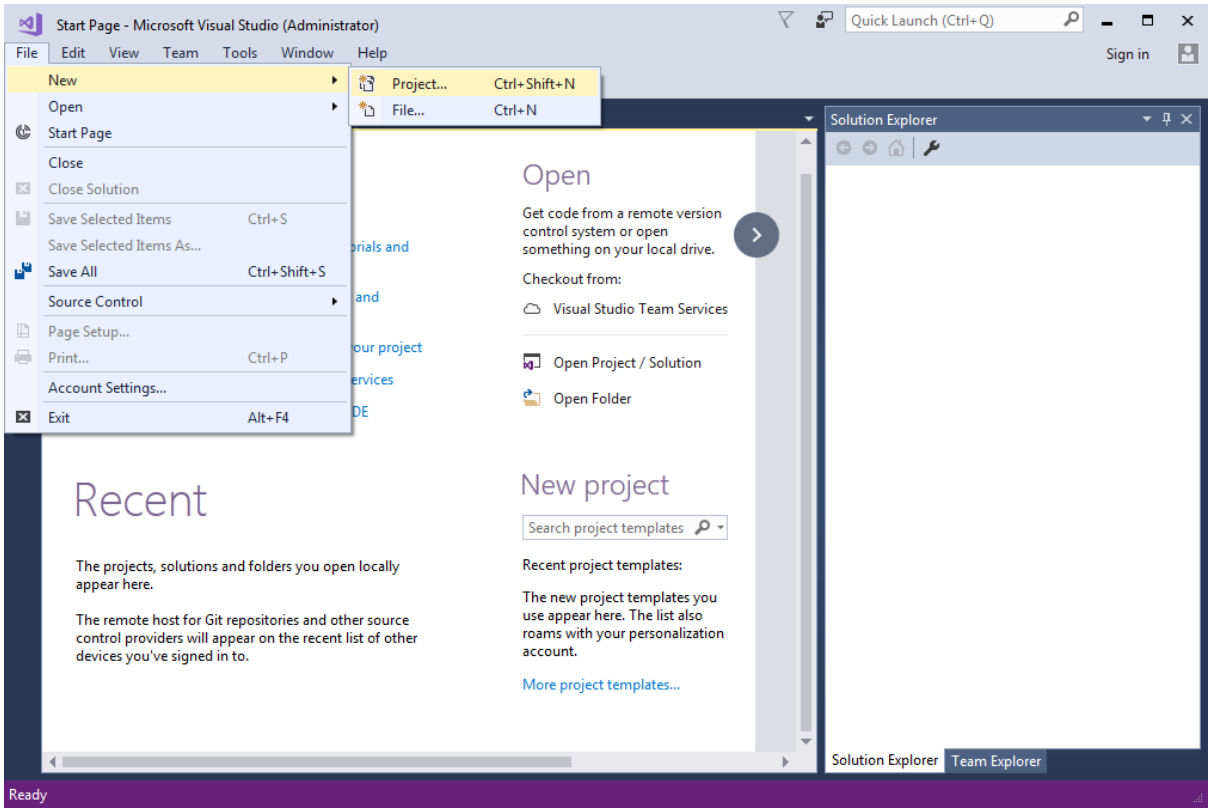
(2) Toolbar به طور معمول شامل همان دستوراتی است که در داخل منوها قرار دارند. Toolbar همانند یک میانبر عمل می‌کند. هر دکمه در Toolbar دارای آیکونی است که کاربرد آنرا نشان می‌دهد. اگر در مورد عملکرد هر کدام از این دکمه‌ها شک داشتید می‌توانید با نشانگر ماوس بر روی آن مکث کوتاهی بکنید تا کاربرد آن به صورت یک پیام (tool tip) نشان داده شود. برخی از دستورات مخفی هستند و تحت شرایط خاص ظاهر می‌شوند. همچنین می‌توانید با کلیک راست بر روی منطقه خالی از Toolbar و یا از مسیر View > Toolbars دستورات بیشتری به آن اضافه کنید. برخی از دکمه‌ها دارای فلش‌های کوچکی هستند که با کلیک بر روی آن‌ها دیگر دستورات وابسته به آن‌ها ظاهر می‌شوند. سمت چپ هر Toolbar به شما اجازه جا به جایی آن را می‌دهد.

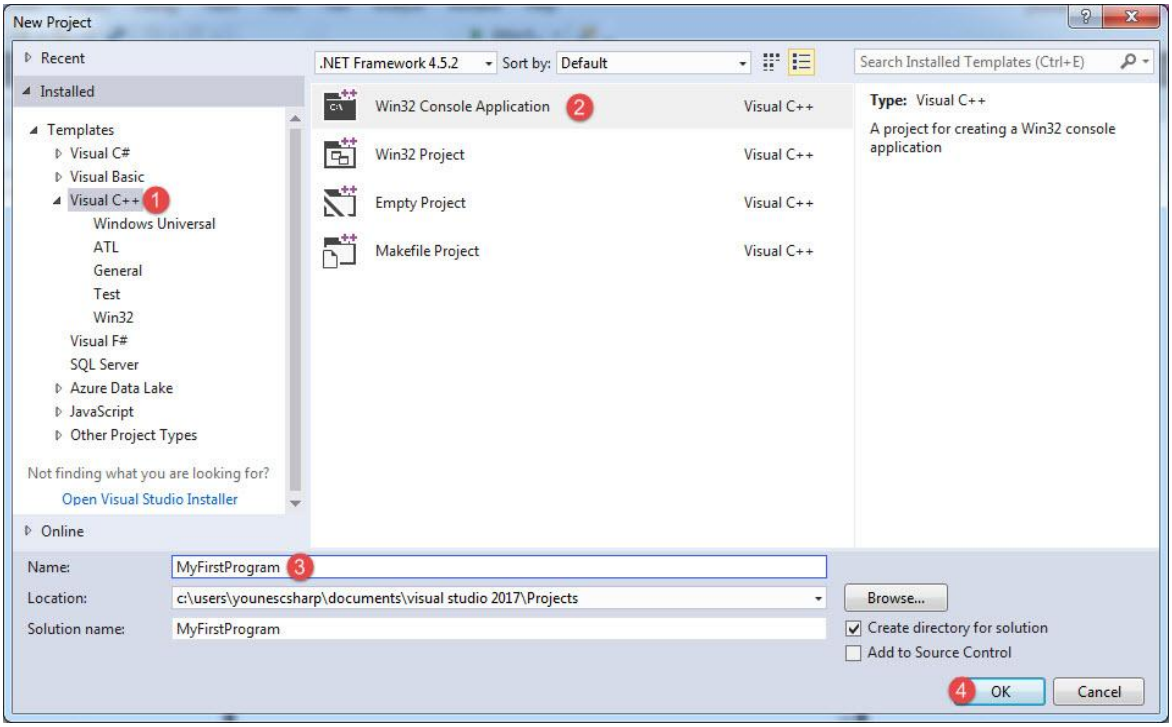
صفحه آغازین (Start Page)

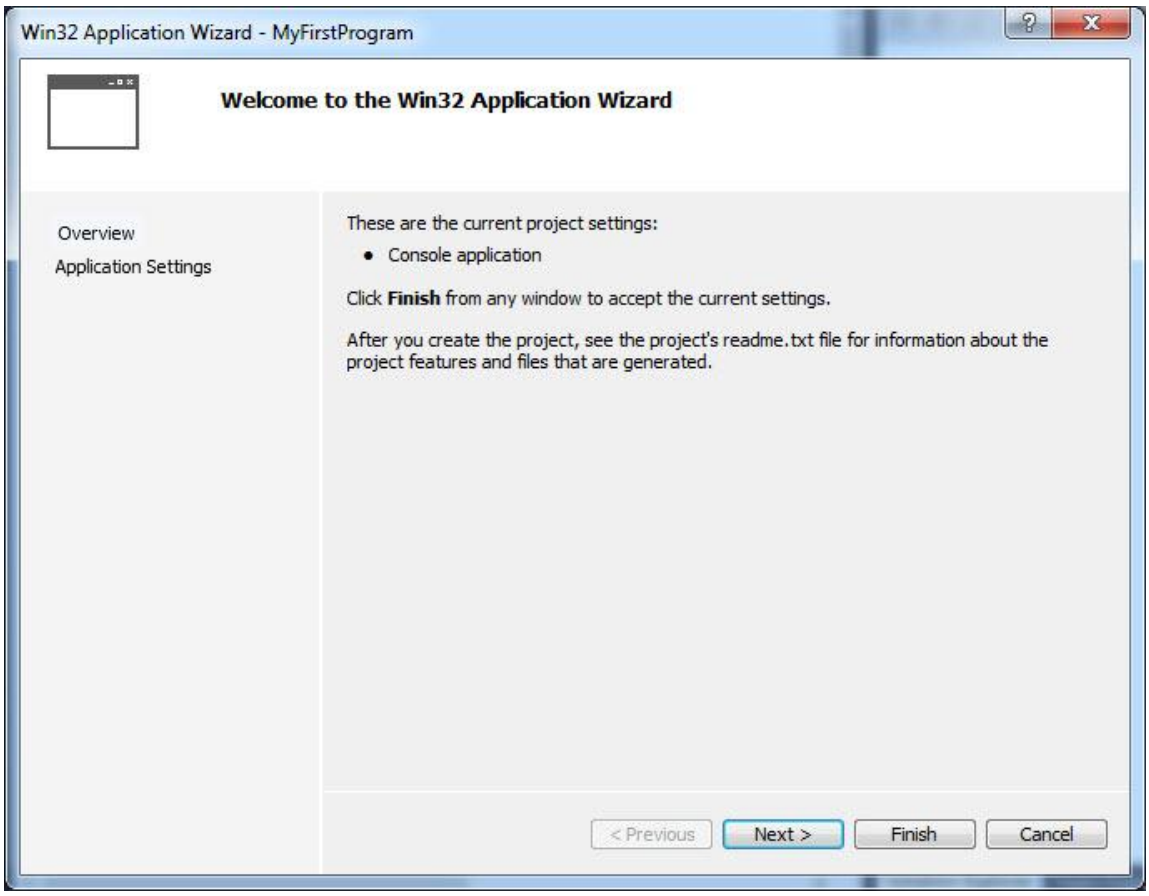
(3) Start برای ایجاد یک پروژه و باز کردن آن از این قسمت استفاده می‌شود. همچنین اگر از قبل پروژه ای ایجاد کرده‌اید می‌توانید آن را در Recent Projects مشاهده و اجرا کنید.

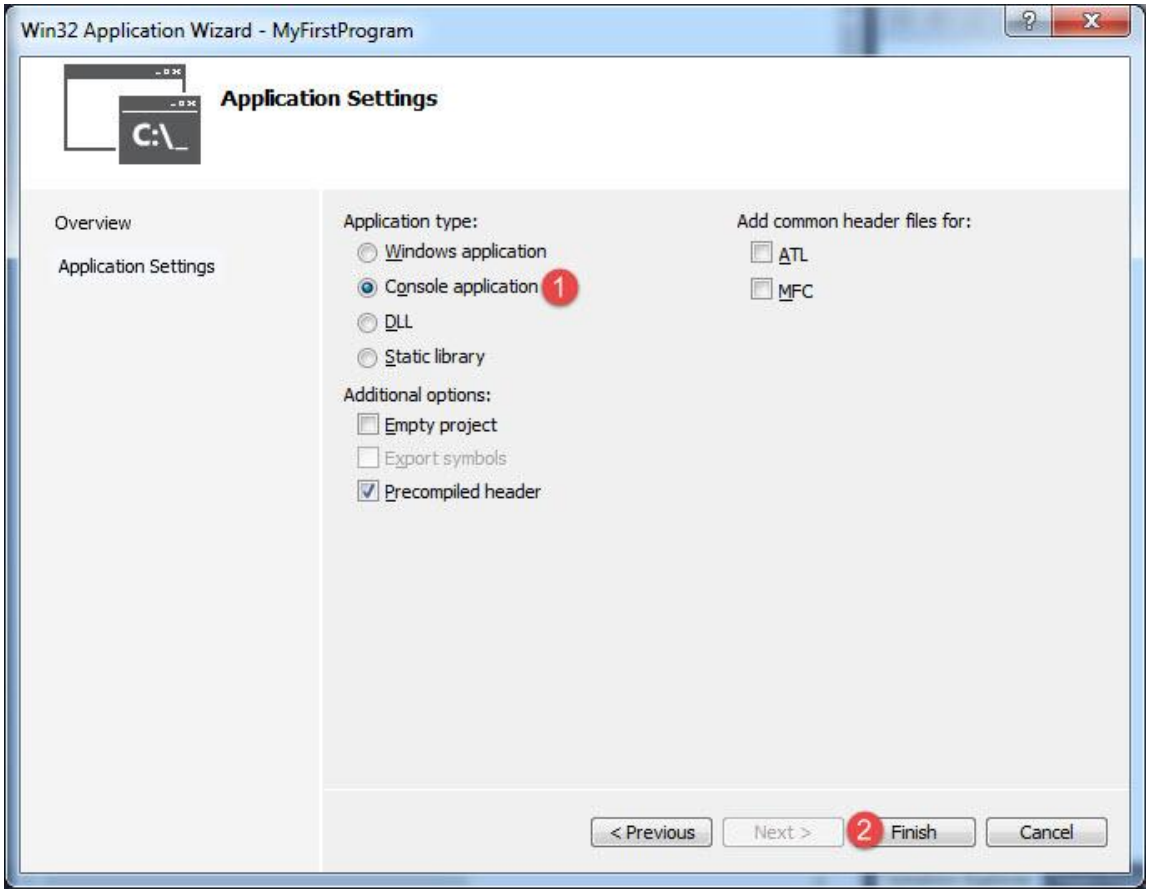
ساخت یک برنامه ساده

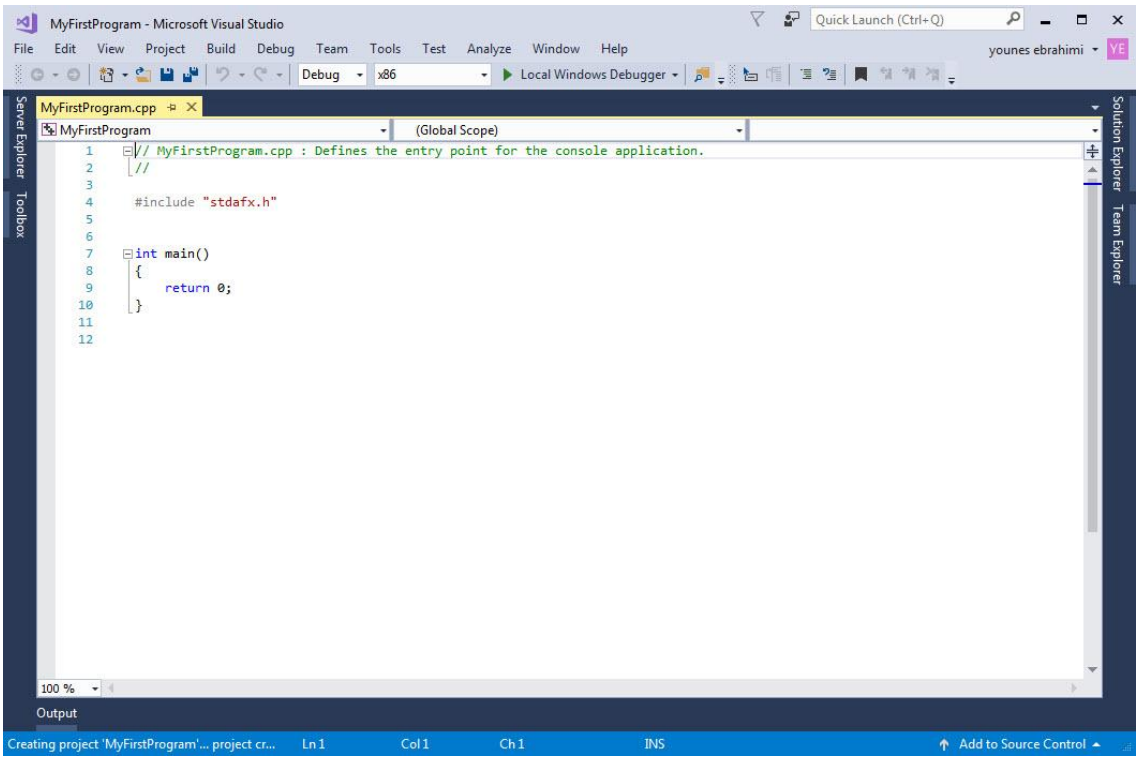
اجازه بدهید یک برنامه بسیار ساده به زبان سی پلاس پلاس (C++) بنویسیم. این برنامه یک پیغام را در محیط کنسول نمایش می‌دهد. در این درس، می‌خواهم ساختار و دستور زبان یک برنامه ساده C++ را توضیح دهم. هر چند که محیط‌های کدنویسی زیادی برای C++ وجود دارند، ولی ما از ساده‌ترین روش برای کدنویسی استفاده می‌کنیم. برنامه ویژوال استودیو را باز کرده و به صورت زیر یک پروژه ایجاد کنید:




New Project
Recent
Installed
Templates
Visual C#
Visual Basic
Visual C++
Windows Universal
ATL
General
Test
Win32
Visual F#
SQL Server
Azure Data Lake
JavaScript
Other Project Types
Not finding what you are looking for?
Open Visual Studio Installer
Online
.NET Framework 4.5.2
Sort by: Default
Search Installed Templates (Ctrl+E)
Win32 Console Application
Win32 Project
Empty Project
Makefile Project
Visual C++
Type: Visual C++
A project for creating a Win32 console application
Name: MyFirstProgram
Location: c:\users\younescsharp\documents\visual studio 2017\Projects
Solution name: MyFirstProgram
Browse...
Create directory for solution
Add to Source Control
OK
Cancel
1
2
3
4


Win32 Application Wizard - MyFirstProgram
Welcome to the Win32 Application Wizard
Overview
Application Settings
These are the current project settings:
Console application
Click Finish from any window to accept the current settings.
After you create the project, see the project's readme.txt file for information about the project features and files that are generated.
< Previous
Next >
Finish
Cancel


Win32 Application Wizard - MyFirstProgram
Application Settings
Overview
Application Settings
Application type:
Windows application
Console application
DLL
Static library
Additional options:
Empty project
Export symbols
Precompiled header
Add common header files for:
ATL
MFC
1
2
< Previous
Next >
Finish
Cancel


MyFirstProgram - Microsoft Visual Studio
Quick Launch (Ctrl+Q)
File Edit View Project Build Debug Team Tools Test Analyze Window Help
younes ebrahimi
Debug
x86
Local Windows Debugger
MyFirstProgram.cpp
MyFirstProgram
(Global Scope)
// MyFirstProgram.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
int main()
{
return 0;
}
Server Explorer
Toolbox
Solution Explorer
Team Explorer
100 %
Output
Creating project 'MyFirstProgram'... project cr...
Ln 1
Col 1
Ch 1
INS
Add to Source Control

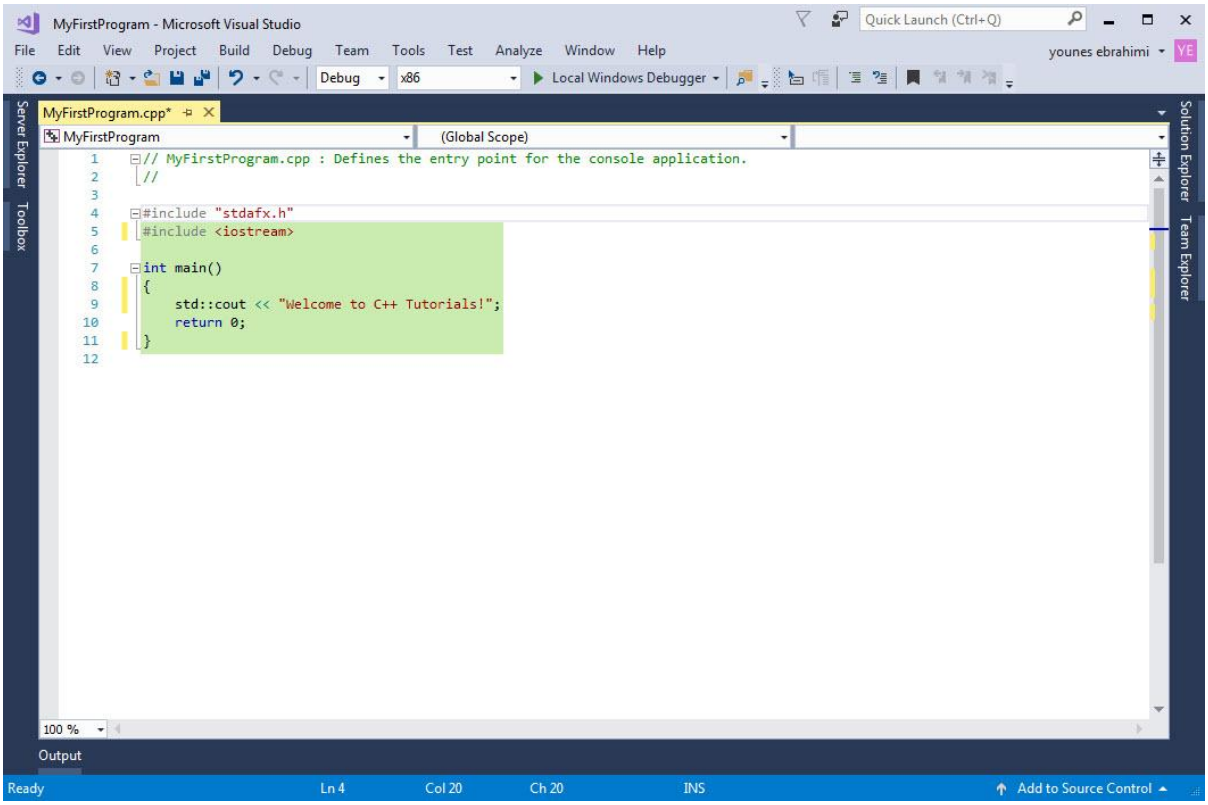
حال کدهای زیر را در این محیط نوشته:

```
#include <iostream>

int main()
{
    std::cout << "Welcome to C++ Tutorials!";

}
```

تا شکل نهایی برنامه به صورت زیر در آید:



ساختار یک برنامه در C++

مثال بالا، ساده‌ترین برنامه‌ای است که شما می‌توانید در C++ بنویسید. هدف در مثال بالا نمایش یک پیغام در صفحه نمایش است. هر زبان برنامه نویسی دارای قواعدی برای کدنویسی است. اجازه بدهید هر خط کد را در مثال بالا توضیح بدهیم. در خطوط ٤ و ٥، فایل هدر یا سرآیند آمده است. فایل‌های سرآیند کتابخانه استاندارد C++ می‌باشند و در این برنامه ما به فایل سرایند iostream نیاز داریم (در درس‌های آینده در مورد این فایل‌ها به طور مفصل توضیح می‌دهیم). خط ۷ متد main() یا متد اصلی

نامیده می‌شود. هر متد شامل یک سری کد است که وقتی اجرا می‌شوند که متد را صدا بزنیم. درباره متد و نحوه صدا زدن آن در فصول بعدی توضیح خواهیم داد. متد main() نقطه آغاز اجرای برنامه است. این بدان معناست که ابتدا تمام کدهای داخل متد main() و سپس بقیه کدها اجرا می‌شود. درباره متد main() در فصول بعدی توضیح خواهیم داد. متد main() و سایر متدها دارای آکولاد و کدهایی در داخل آن‌ها می‌باشند و وقتی کدها اجرا می‌شوند که متدها را صدا بزنیم. هر خط کد در C++ به یک سمیکالن (;) ختم می‌شود. اگر سمیکالن در آخر خط فراموش شود برنامه با خطا مواجه می‌شود. مثالی از یک خط کد در C++ به صورت زیر است:

```
std::cout << "Welcome to C++ Tutorials!";
```

این خط کد پیغام Welcome to Visual C++ Tutorials! را در صفحه نمایش نشان می‌دهد. از شیء cout برای چاپ یک رشته استفاده می‌شود. یک رشته گروهی از کاراکترها است، که به وسیله دابل کوتیشن (") محصور شده است. مانند "Welcome to Visual C++ Tutorials!".

یک کاراکتر می‌تواند یک حرف، عدد، علامت یا ... باشد. در کل مثال بالا نحوه استفاده از شیء cout است که در داخل فضای نام std قرار دارد را نشان می‌دهد. توضیحات بیشتر در درسهای آینده آمده است. C++ فضای خالی و خطوط جدید را نادیده می‌گیرد. بنابراین شما می‌توانید همه برنامه را در یک خط بنویسید. اما اینکار خواندن و اشکال زدایی برنامه را مشکل می‌کند. یکی از خطاهای معمول در برنامه نویسی فراموش کردن سمیکالن در پایان هر خط کد است. به مثال زیر توجه کنید:

```
std::cout <<
"Welcome to C++ Tutorials!";
```

سی پلاس پلاس فضای خالی بالا را نادیده می‌گیرد و از کد بالا اشکال نمی‌گیرد. اما از کد زیر ایراد می‌گیرد:

```
std::cout << ;
"Welcome to C++ Tutorials!";
```

به سمیکالن آخر خط اول توجه کنید. برنامه با خطای نحوی مواجه می‌شود چون دو خط کد مربوط به یک برنامه هستند و شما فقط باید یک سمیکالن در آخر آن قرار دهید. همیشه به یاد داشته باشید که C++ به بزرگی و کوچکی حروف حساس است. یعنی به طور مثال MAN و man در سی پلاس پلاس با هم فرق دارند. رشته‌ها و توضیحات از این قاعده مستثنی هستند که در درسهای آینده توضیح خواهیم داد. مثلاً کدهای زیر با خطا مواجه می‌شوند و اجرا نمی‌شوند:

```
std::cOUT << "Welcome to C++ Tutorials!";
STD::cout << "Welcome to C++ Tutorials!";
Std::Cout << "Welcome to C++ Tutorials!";
```

تغییر در بزرگی و کوچکی حروف از اجرای کدها جلوگیری می‌کند. اما کد زیر کاملاً بدون خطا است:

```
std::cout << "Welcome to C++ Tutorials!";
```

همیشه کدهای خود را در داخل آکولاد بنویسید.

```
{
    statement1;
}
```

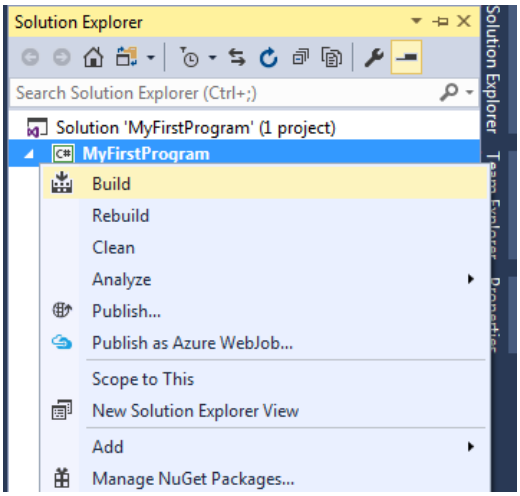
این کار باعث می‌شود که کدنویسی شما بهتر به چشم بیاید و تشخیص خطاها راحت‌تر باشد.

ذخیره پروژه و برنامه

برای ذخیره پروژه و برنامه می‌توانید به مسیر File > Save All بروید یا از کلیدهای میانبر Ctrl+Shift+S استفاده کنید. همچنین می‌توانید از قسمت Toolbar بر روی شکل  کلیک کنید. برای ذخیره یک فایل ساده می‌توانید به مسیر File > Save (FileName) بروید یا از کلیدهای میانبر Ctrl+S استفاده کنید. همچنین می‌توانید از قسمت Toolbar بر روی شکل  کلیک کنید. برای باز کردن یک پروژه یا برنامه از منوی File گزینه Open را انتخاب می‌کنید یا بر روی آیکون  در toolbar کلیک کنید. سپس به محلی که پروژه در آنجا ذخیره شده می‌روید و فایلی با پسوند sln یا پروژه با پسوند csproj. را باز می‌کنید.

کامپایل برنامه

برای کامپایل برنامه از منوی Debug گزینه Build Solution را انتخاب می‌کنید یا دکمه F6 را بر روی صفحه کلید فشار می‌دهیم. این کار همه پروژه‌های داخل solution را کامپایل می‌کند. برای کامپایل یک قسمت از solution به Solution Explorer می‌رویم و بر روی آن قسمت راست کلیک کرده و از منوی باز شوند گزینه build را انتخاب می‌کنید. مانند شکل زیر:



اجرای برنامه

دو راه برای اجرای برنامه وجود دارد:

- اجرا همراه با اشکال زدایی(Debug)
- اجرا بدون اشکال زدایی(Non-Debug)

اجرای بدون اشکال زدایی برنامه، خطاهای برنامه را نادیده می‌گیرد. با اجرای برنامه در حالت Non-Debug سریعاً برنامه اجرا می‌شود و شما با زدن یک دکمه از برنامه خارج می‌شوید. در حالت پیش فرض حالت Non-Debug مخفی است و برای استفاده از آن می‌توان از منوی Debug گزینه Start Without Debuging را انتخاب کرد یا از دکمه‌های ترکیبی Crl + F5 استفاده نمود:

```
Welcome to C++ Tutorials!Press any key to continue...
```

به این نکته توجه کنید که پیغام Press any key to continue… جز خروجی به حساب نمی‌آید و فقط نشان دهنده آن است که برنامه در حالت Non-Debug اجرا شده است و شما می‌توانید با زدن یک کلید از برنامه خارج شوید. برای اینکه تفکیکی بین عبارت مورد نظر ما و عبارت به وجود بیاید کافیست که خط ۹ کد ابتدای درس را به صورت زیر تغییر دهید:

```
std::cout << "Welcome to C++ Tutorials!" << endl;
```

حال اگر برنامه را دوباره اجرا کنید، خروجی به صورت زیر نمایش داده می شود:

```
Welcome to C++ Tutorials!
Press any key to continue...
```

دسترسی به حالت Debug Mode آسان تر است و به صورت پیشفرض برنامه‌ها در این حالت اجرا می‌شوند. از این حالت برای رفع خطاها و اشکال زدایی برنامه‌ها استفاده می‌شود که در درس‌های آینده توضیح خواهیم داد. شما همچنین می‌توانید از break points و قسمت Help برنامه در مواقعی که با خطا مواجه می‌شوید استفاده کنید. برای اجرای برنامه با حالت Debug Mode می‌توانید از منوی Debug گزینه Start Debugging را انتخاب کرده و یا دکمه F5 را فشار دهید. همچنین می‌توانید بر روی شکل 

در toolbar کلیلک کنید. اگر از حالت Debug Mode استفاده کنید برنامه نمایش داده شده و فوراً ناپدید می‌شود. برای جلوگیری از این اتفاق شما می‌توانید از کلاس و متد std::cin.get(); قبل از عبارت return 0، برای توقف برنامه و گرفتن ورودی از کاربر جهت خروج از برنامه استفاده کنید (درباره متدها در درس‌های آینده توضیح خواهیم داد):

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

int main()
{
    std::cout << "Welcome to C++ Tutorials!" << endl;
    std::cin.get();
    return 0;
}
```

به این نکته توجه کنید که در درس های بعدی خط ۱ کد بالا را حذف نکرده و از این خط به بعد کدهای خود را بنویسید.

وارد کردن فضای نام در برنامه

در برنامه فوق ما یک فضای نام در برنامه‌مان با نام std داریم، اما سی پلاس پلاس دارای تعداد زیادی فضای نام می‌باشد. یکی از این فضاهای نامی، فضای نام std است که شیء cout که ما از آن در برنامه بالا استفاده کردیم در این فضای نام قرار دارد.

```
std::cout << "Welcome to C++ Tutorials!" << endl;
```

اینکه قبل از استفاده از هر کلاس ابتدا فضای نام آن را مانند کد بالا بنویسیم کمی خسته کننده است. خوشبختانه C++ به ما اجازه می‌دهد که برای جلوگیری از تکرار مکررات، فضاهای نامی را که قرار است در برنامه استفاده کنیم با استفاده از دستور using و کلمه namespace در ابتدای برنامه وارد نماییم:

```
using namespace NameofNameSpace;
```

دستور بالا نحوه وارد کردن یک فضای نام در برنامه را نشان می‌دهد. در نتیجه به جای آنکه به صورت زیر ابتدا نام فضای نام و سپس نام کلاس را بنویسیم:

```
std::cout << "Welcome to C++ Tutorials!" << endl;
```

می‌توانیم فضای نام را با دستوری که ذکر شد وارد برنامه کرده و کد بالا را به صورت خلاصه شده زیر بنویسیم:

```
cout << "Welcome to C++ Tutorials!" << endl;
```

دستورات `using` که باعث وارد شدن فضاهای نامی به برنامه می‌شوند عموماً در ابتدای برنامه و قبل از همه کدها نوشته می‌شوند، پس برنامه‌ی این درس را می‌توان به صورت زیر نوشت:

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    cout << "Welcome to C++ Tutorials!" << endl;

}
```

حال که با خصوصیات و ساختار اولیه `C++` آشنا شدید در درسهای آینده مطالب بیشتری از این زبان برنامه نویسی قدرتمند خواهید آموخت.

توضیحات

وقتی که کدی تایپ می‌کنید شاید بخواهید که متنی جهت یادآوری وظیفه آن کد به آن اضافه کنید. در `C++` (و بیشتر زبانهای برنامه نویسی) می‌توان این کار را با استفاده از توضیحات انجام داد. توضیحات متونی هستند که توسط کامپایلر نادیده گرفته می‌شوند و به عنوان بخشی از کد محسوب نمی‌شوند.

هدف اصلی از ایجاد توضیحات، بالا بردن خوانایی و تشخیص نقش کدهای نوشته شده توسط شما، برای دیگران است. فرض کنید که می‌خواهید در مورد یک کد خاص، توضیح بدهید، می‌توانید توضیحات را در بالای کد یا کنار آن بنویسید. از توضیحات برای مستند سازی برنامه هم استفاده می‌شود. در برنامه زیر نقش توضیحات نشان داده شده است:

```cpp
#include <iostream>
using namespace std;

int main()
{
    // This line will print the message hello world
    cout << "Hello World!";
}
```

در کد بالا، خط ٦ یک توضیح درباره خط ۷ است که به کاربر اعلام می‌کند که وظیفه خط ۷ چیست؟ با اجرای کد بالا فقط جمله Hello World چاپ شده و خط ۷ در خروجی نمایش داده نمی‌شود چون کامپایلر توضیحات را نادیده می‌گیرد. توضیحات بر دو نوع‌اند:

توضیحات تک خطی

```cpp
// single line comment
```

توضیحات چند خطی

```cpp
/* multi
 line
 comment */
```

توضیحات تک خطی همانگونه که از نامش پیداست، برای توضیحاتی در حد یک خط به کار می‌روند. این توضیحات با علامت // شروع می‌شوند و هر نوشته‌ای که در سمت راست آن قرار بگیرد جز توضیحات به حساب می‌آید. این نوع توضیحات معمولاً در بالا یا کنار کد قرار می‌گیرند. اگر توضیح درباره یک کد به بیش از یک خط نیاز باشد از توضیحات چند خطی استفاده می‌شود. توضیحات چند خطی با /* شروع و با */ پایان می‌یابند. هر نوشته‌ای که بین این دو علامت قرار بگیرد جز توضیحات محسوب می‌شود.

کاراکترهای کنترلی

کاراکترهای کنترلی کاراکترهای ترکیبی هستند که با یک بک اسلش (\) شروع می‌شوند و به دنبال آن‌ها یک حرف یا عدد می‌آید و یک رشته را با فرمت خاص نمایش می‌دهند. برای مثال برای ایجاد یک خط جدید و قرار دادن رشته در آن می‌توان از کاراکتر کنترلی \n استفاده کرد:

```cpp
#include <iostream>
using namespace std;
```

```
int main()
{
    cout << "Hello\nWorld";
}
```

```
Hello
World
```

مشاهده کردید که کامپایلر بعد از مواجهه با کاراکتر کنترلی \n نشانگر ماوس را به خط بعد برده و بقیه رشته را در خط بعد نمایش می‌دهد. جدول زیر لیست کاراکترهای کنترلی و کارکرد آن‌ها را نشان می‌دهد:

عملکرد	کاراکتر کنترلی	عملکرد	کاراکتر کنترلی
Form Feed	\f	چاپ کوتیشن	\’
خط جدید	\n	چاپ دابل کوتیشن	\”
سر سطر رفتن	\r	چاپ بک اسلش	\\
حرکت به صورت افقی	\t	چاپ فضای خالی	\0
حرکت به صورت عمودی	\v	صدای بیپ	\a
چاپ کاراکتر یونیکد	\u	حرکت به عقب	\b

ما برای استفاده از کاراکترهای کنترلی از بک اسلش (\) استفاده می‌کنیم. از آنجاییکه \ معنای خاصی به رشته‌ها می‌دهد برای چاپ بک اسلش (\) باید از (\\) استفاده کنیم:

```
cout << "We can print a \\ by using the \\\\ escape sequence.";
```

```
We can print a \ by using the \\ escape sequence.
```

یکی از موارد استفاده از \\، نشان دادن مسیر یک فایل در ویندوز است:

```
cout << "C:\\Program Files\\Some Directory\\SomeFile.txt";
```

```
C:\Program Files\Some Directory\SomeFile.txt
```

از آنجاییکه از دابل کوتیشن (“) برای نشان دادن رشته‌ها استفاده می‌کنیم برای چاپ آن از ”\استفاده می‌کنیم:

```
cout << "I said, \"Motivate yourself!\".";
```

```
I said, "Motivate yourself!".
```

همچنین برای چاپ کوتیشن (') از \' استفاده می‌کنیم:

```
cout << "The programmer\'s heaven.";
```

```
The programmer's heaven.
```

برای ایجاد فاصله بین حروف یا کلمات از \t استفاده می‌شود:

```
cout << "Left\tRight";
```

```
Left	Right
```

هر تعداد کاراکتر که بعد از کاراکتر کنترلی \r بیایند به اول سطر منتقل و جایگزین کاراکترهای موجود می‌شوند:

```
cout << "Mitten\rK";
```

```
Kitten
```

مثلاً در مثال بالا کاراکتر K بعد از کاراکتر کنترلی \r آمده است. کاراکتر کنترلی حرف K را به ابتدای سطر برده و جایگزین حرف M می‌کند. برای چاپ کاراکترهای یونیکد می‌توان از \u استفاده کرد. برای استفاده از \u، مقدار در مبنای ۱۶ کاراکتر را درست بعد از علامت \u قرار می‌دهیم. برای مثال اگر بخواهیم علامت (©) را چاپ کنیم باید بعد از علامت \u مقدار 00A9 را قرار دهیم مانند :

```
cout << "\u00A9";
```

```
©
```

برای مشاهده لیست مقادیر مبنای ۱۶ برای کاراکترهای یونیکد به لینک زیر مراجعه نمایید:

http://www.ascii.cl/htmlcodes.htm

اگر کامپایلر به یک کاراکتر کنترلی غیر مجاز برخورد کند، برنامه پیغام خطا می‌دهد. بیشترین خطا زمانی اتفاق می‌افتد که برنامه نویس برای چاپ اسلش (\) از \\ استفاده می‌کند. برای دریافت اطلاعات بیشتر در مورد کاراکترهای کنترلی به لینک زیر مراجعه کنید:

https://msdn.microsoft.com/en-us/library/h21280bw.aspx

متغیر

متغیر مکانی از حافظه است که شما می‌توانید مقادیری را در آن ذخیره کنید. می‌توان آن را به عنوان یک ظرف تصور کرد که داده‌های خود را در آن قرار داده‌اید. محتویات این ظرف می‌تواند پاک شود یا تغییر کند. هر متغیر دارای یک نام نیز هست. که از طریق آن می‌توان متغیر را از دیگر متغیرها تشخیص داد و به مقدار آن دسترسی پیدا کرد. همچنین دارای یک مقدار می‌باشد که می‌تواند توسط کاربر انتخاب شده باشد یا نتیجه یک محاسبه باشد. مقدار متغیر می‌تواند تهی نیز باشد. متغیر دارای نوع نیز هست بدین معنی که نوع آن با نوع داده‌ای که در آن ذخیره می‌شود یکی است. متغیر دارای عمر نیز هست که از روی آن می‌توان تشخیص داد که متغیر باید چقدر در طول برنامه مورد استفاده قرار گیرد. و در نهایت متغیر دارای محدوده استفاده نیز هست که به شما می‌گوید که متغیر در چه جای برنامه برای شما قابل دسترسی است. ما از متغیرها به عنوان یک انبار موقتی برای ذخیره داده استفاده می‌کنیم. هنگامی که یک برنامه ایجاد می‌کنیم احتیاج به یک مکان برای ذخیره داده، مقادیر یا داده‌هایی که توسط کاربر وارد می‌شوند داریم. ایم مکان همان متغیر است. برای این از کلمه متغیر استفاده می‌شود چون ما می‌توانیم بسته به نوع شرایط هر جا که لازم باشد مقدار آن را تغییر دهیم. متغیرها موقتی هستند و فقط موقعی مورد استفاده قرار می‌گیرند که برنامه در حال اجراست و وقتی شما برنامه را می‌بندید محتویات متغیرها نیز پاک می‌شود. قبلاً ذکر شد که به وسیله نام متغیر می‌توان به آن دسترسی پیدا کرد. برای نامگذاری متغیرها باید قوانین زیر را رعایت کرد:

- نام متغیر باید با یک از حروف الفبا (a-z or A-Z) شروع شود.
- نمی‌تواند شامل کاراکترهای غیرمجاز مانند #، ?، ^، $ و . باشد.
- نمی‌توان از کلمات رزرو شده در C++ برای نام متغیر استفاده کرد.
- نام متغیر نباید دارای فضای خالی (spaces) باشد.
- اسامی متغیرها نسبت به بزرگی و کوچکی حروف حساس هستند. در C++ دو حرف مانند a و A دو کاراکتر مختلف به حساب می‌آیند.

دو متغیر با نامهای myNumber و MyNumber دو متغیر مختلف محسوب می‌شوند چون یکی از آن‌ها با حرف کوچک m و دیگری با حرف بزرگ M شروع می‌شود. شما نمی‌توانید دو متغیر را که دقیق شبیه هم هستند را در یک scope (محدوده) تعریف کنید. Scope به معنای یک بلوک کد است که متغیر در آن قابل دسترسی و استفاده است. در مورد Scope در فصل‌های آینده بیشتر توضیح خواهیم داد. متغیر دارای نوع هست که نوع داده‌ای را که در خود ذخیره می‌کند را نشان می‌دهد. معمول‌ترین انواع داده int، double، string، char و float می‌باشند. برای مثال شما برای قرار دادن یک عدد صحیح در متغیر باید از نوع int استفاده کنید.

انواع ساده

انواع ساده انواعی از داده‌ها هستند که شامل اعداد، کاراکترها و رشته‌ها و مقادیر بولی می‌باشند. به انواع ساده انواع اصلی نیز گفته می‌شود چون از آن‌ها برای ساخت انواع پیچیده‌تری مانند کلاس‌ها و ساختارها استفاده می‌شود. انواع ساده دارای مجموعه مشخصی از مقادیر هستند و محدوده خاصی از اعداد را در خود ذخیره می‌کنند. در C++ هفت نوع داده وجود دارد که در جدول زیر ذکر شده‌اند:

کلمه کلیدی	نوع
bool	Boolean
char	Character
int	Integer
float	Floating point
double	Double floating point
void	Valueless
wchar_t	Wide character

انواع بالا (به جز void) می‌توانند با عباراتی مثل signed، long، unsigned و short ترکیب شده و نوع‌های دیگری را به وجود آورند:

نوع	مقدار فضایی که از حافظه اشغال می‌کند	محدوده
char	1byte	-128 to 127 or 0 to 255
unsigned char	1byte	0 to 255
signed char	1byte	-128 to 127
int	4bytes	-2147483648 to 2147483647
unsigned int	4bytes	0 to 4294967295
signed int	4bytes	-2147483648 to 2147483647
short int	2bytes	-32768 to 32767
unsigned short int	2bytes	0 to 65,535

signed short int	2bytes	-32768 to 32767
long int	8bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
signed long int	8bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
unsigned long int	8bytes	0 to 18,446,744,073,709,551,615
float	4bytes	+/- 3.4e +/- 38 (~7 digits)
double	8bytes	+/- 1.7e +/- 308 (~15 digits)
long double	8bytes	+/- 1.7e +/- 308 (~15 digits)
wchar_t	2 or 4 bytes	1 wide character

نوع char برای ذخیره کاراکترهای یونیکد استفاده می‌شود. کاراکترها باید داخل یک کوتیشن ساده قرار بگیرند مانند ('a').

نوع bool فقط می‌تواند مقادیر درست (true) یا نادرست (false) را در خود ذخیره کند و بیشتر در برنامه‌هایی که دارای ساختار تصمیم گیری هستند مورد استفاده قرار می‌گیرد. نوع string برای ذخیره گروهی از کاراکترها مانند یک پیغام استفاده می‌شود. مقادیر ذخیره شده در یک رشته باید داخل دابل کوتیشن قرار گیرند تا توسط کامپایلر به عنوان یک رشته در نظر گرفته شوند مانند ("massage").

استفاده از متغیرها

در مثال زیر نحوه تعریف و مقدار دهی متغیرها نمایش داده شده است:

```
#include <iostream>
using namespace std;

int main()
{
    //Declare variables
    int num1;
    int num2;
    double num3;
    double num4;
    bool boolVal;
    char myChar;
    string message;
```

```
    //Assign values to variables
    num1 = 1;
    num2 = 2;
    num3 = 3.54;
    num4 = 4.12;
    boolVal = true;
    myChar = 'R';
    message = "Hello World!";

    //Show the values of the variables
    cout << "num1 = " << num1 << endl;
    cout << "num2 = " << num2 << endl;
    cout << "num3 = " << num3 << endl;
    cout << "num4 = " << num4 << endl;
    cout << "boolVal = " << boolVal << endl;
    cout << "myChar = " << myChar << endl;
    cout << "message = " << message << endl;

}
```

تعریف متغیر

در کد بالا متغیرهایی با نوع و نام متفاوت تعریف شده‌اند. ابتدا باید نوع داده‌هایی را که این متغیرها قرار است در خود ذخیره کنند را مشخص کنیم و سپس یک نام برای آن‌ها در نظر بگیریم و در آخر سیمیکولن بگذاریم. همیشه به یاد داشته باشید که قبل از مقدار دهی و استفاده از متغیر باید آن را تعریف کرد. شاید برایتان این سؤال پیش آمده باشد که کاربرد `endl` چیست؟ `endl` برای ایجاد خط جدید مورد استفاده قرار گرفته است. یعنی نشانگر ماوس را همانند کاراکتر کنترلی `\n` به خط بعد می‌برد، در نتیجه خروجی کد بالا در خطوط جداگانه چاپ می‌شود .

```
//Declare variables
int num1;
int num2;
double num3;
double num4;
bool boolVal;
char myChar;
string message;
```

نحوه تعریف متغیر به صورت زیر است:

```
data_type identifier;
```

date_type همان نوع داده است مانند int، double و .... dentifier نیز نام متغیر است که به ما امکان استفاده و دسترسی به مقدار متغیر را می‌دهد. برای تعریف چند متغیر از یک نوع می‌توان به صورت زیر عمل کرد:

```
data_type identifier1, identifier2, ... indentifierN;
```

مثال

```
int num1, num2, num3, num4, num5;
string message1, message2, message3;
```

در مثال بالا ۵ متغیر از نوع صحیح و ۳ متغیر از نوع رشته تعریف شده است. توجه داشته باشید که بین متغیرها باید علامت کاما (,) باشد.

نامگذاری متغیرها

- نام متغیر باید با یک حرف یا زیرخط و به دنبال آن حرف یا عدد شروع شود.
- نمی‌توان از کاراکترهای خاص مانند #، %، & یا عدد برای شروع نام متغیر استفاده کرد مانند 2numbers.
- نام متغیر نباید دارای فاصله باشد. برای نام‌های چند حرفی می‌توان به جای فاصله از علامت زیرخط یا _ استفاده کرد.

نامهای مجاز:

```
num1  myNumber  studentCount  total       first_name    _minimum
num2  myChar    average      amountDue   last_name     _maximum
name  counter   sum          isLeapYear  color_of_car  _age
```

نامهای غیر مجاز:

```
123        #numbers#  #ofstudents  1abc2
123abc     $money     first name   ty.np
my number  this&that  last name    1:00
```

اگر به نامهای مجاز در مثال بالا توجه کنید متوجه قراردادهای به کار رفته در نامگذاری آن‌ها خواهید شد. یکی از روش‌های نامگذاری، نامگذاری کوهان شتری است. در این روش که برای متغیرهای دو کلمه‌ای به کار می‌رود، اولین حرف اولین کلمه با حرف کوچک و اولین حرف دومین کلمه با حرف بزرگ نمایش داده می‌شود مانند myNumber. توجه کنید که اولین حرف کلمه Number با حرف بزرگ شروع شده است. مثال دیگر کلمه numberOfStudents است. اگز توجه کنید بعد از اولین کلمه حرف اول سایر کلمات با حروف بزرگ نمایش داده شده است.

محدوده متغیر

متغیرهای ابتدای درس در داخل متد main() تعریف شده‌اند. در نتیجه این متغیرها فقط در داخل متد main() قابل دسترسی هستند. محدوده یک متغیر مشخص می‌کند که متغیر در کجای کد قابل دسترسی است. هنگامیکه برنامه به پایان متد main() می‌رسد متغیرها از محدوده خارج و بدون استفاده می‌شوند تا زمانی که برنامه در حال اجراست. محدوده متغیرها انواعی دارد که در درس‌های بعدی با آن‌ها آشنا می‌شوید. تشخیص محدوده متغیر بسیار مهم است چون به وسیله آن می‌فهمید که در کجای کد می‌توان از متغیر استفاده کرد. باید یاد آور شد که دو متغیر در یک محدوده نمی‌توانند دارای نام یکسان باشند. مثلاً کد زیر در برنامه ایجاد خطا می‌کند:

```
int num1;
int num1;
```

از آنجاییکه C++ به بزرگی و کوچک بودن حروف حساس است می‌توان از این خاصیت برای تعریف چند متغیر هم نام ولی با حروف متفاوت (از لحاظ بزرگی و کوچکی) برای تعریف چند متغیر از یک نوع استفاده کرد مانند:

```
int num1;
int Num1;
int NUM1;
```

مقداردهی متغیرها

می‌توان فوراً بعد از تعریف متغیرها مقادیری را به آن‌ها اختصاص داد. این عمل را مقداردهی می‌نامند. در زیر نحوه مقدار دهی متغیرها نشان داده شده است:

```
data_type identifier = value;
```

به عنوان مثال:

```
int myNumber = 7;
```

همچنین می‌توان چندین متغیر را فقط با گذاشتن کاما بین آن‌ها به سادگی مقدار دهی کرد:

```
data_type variable1 = value1, varaible2 = value2, ... variableN, valueN;
int num1 = 1, num2 = 2, num3 = 3;
```

تعریف متغیر با مقدار دهی متغیرها متفاوت است. تعریف متغیر یعنی انتخاب نوع و نام برای متغیر ولی مقدار دهی یعنی اختصاص یک مقدار به متغیر.

اختصاص مقدار به متغیر

در زیر نحوه اختصاص مقادیر به متغیرها نشان داده شده است:

```
num1 = 1;
num2 = 2;
num3 = 3.54;
num4 = 4.12;
boolVal = true;
myChar = 'R';
message = "Hello World!";
```

به این نکته توجه کنید که شما به مغیری که هنوز تعریف نشده نمی‌توانید مقدار بدهید. شما فقط می‌توانید از متغیرهایی استفاده کنید که هم تعریف و هم مقدار دهی شده باشند. مثلاً متغیرهای بالا همه قابل استفاده هستند. در این مثال num1 و num2 هر دو تعریف شده‌اند و مقادیری از نوع صحیح به آن‌ها اختصاص داده شده است. اگر نوع داده با نوع متغیر یکی نباشد برنامه پیغام خطا می‌دهد.

ثابت

ثابت‌ها انواعی هستند که مقدار آن‌ها در طول برنامه تغییر نمی‌کند. ثابت‌ها حتماً باید مقدار دهی اولیه شوند و اگر مقدار دهی آن‌ها فراموش شود در برنامه خطا به وجود می‌آید. بعد از این که به ثابت‌ها مقدار اولیه اختصاص داده شد هرگز در زمان اجرای برنامه نمی‌توان آن را تغییر داد. برای تعریف ثابت‌ها باید از کلمه کلیدی const و #define استفاده کرد. معمولاً نام ثابت‌ها را طبق قرارداد با حروف بزرگ می‌نویسند تا تشخیص آن‌ها در برنامه راحت باشد. نحوه تعریف ثابت در زیر آمده است:

```
const data_type identifier = initial_value;
```

در کد بالا ابتدا کلمه کلیدی const و سپس نوع ثابت و بعد نام ثابت را با حروف بزرگ می‌نویسیم. و در نهایت یک مقدار را به آن اختصاص می‌دهیم و علامت سمیکالن می‌گذاریم.

```
#define data_type identifier initial_value
```

در روش بالا فقط #define را نوشته و سپس نام ثابت و بعد مقداری که قرار است دریافت کند. به این نکته توجه کنید که در روش بالا نه علامت سمیکالن وجود دارد و نه علامت مساوی. مثال:

```
#include <iostream>
using namespace std;

int main()
```

```
{
    const int NUMBER = 1;

    NUMBER = 20; //ERROR, Cant modify a constant

    cout << NUMBER;
}
```

```
#include <iostream>
using namespace std;

int main()
{
    #define NUMBER 1

    NUMBER = 20; //ERROR, Cant modify a constant

    cout << NUMBER;
}
```

در این مثال می‌بینید که مقدار دادن به یک ثابت، که قبلاً مقدار دهی شده برنامه را با خطا مواجه می‌کند. نکتهٔ دیگری که نباید فراموش شود این است که نباید مقدار ثابت را با مقدار دیگر متغیرهای تعریف شده در برنامه برابر قرار داد.

مثال:

```
int someVariable;
const int MY_CONST = someVariable;
```

ممکن است این سؤال برایتان پیش آمده باشد که دلیل استفاده از ثابت‌ها چیست؟ اگر مطمئن هستید که مقادیری در برنامه وجود دارند که هرگز در طول برنامه تغییر نمی‌کنند بهتر است که آن‌ها را به صورت ثابت تعریف کنید. این کار هر چند کوچک کیفیت برنامه شما را بالا می‌برد.