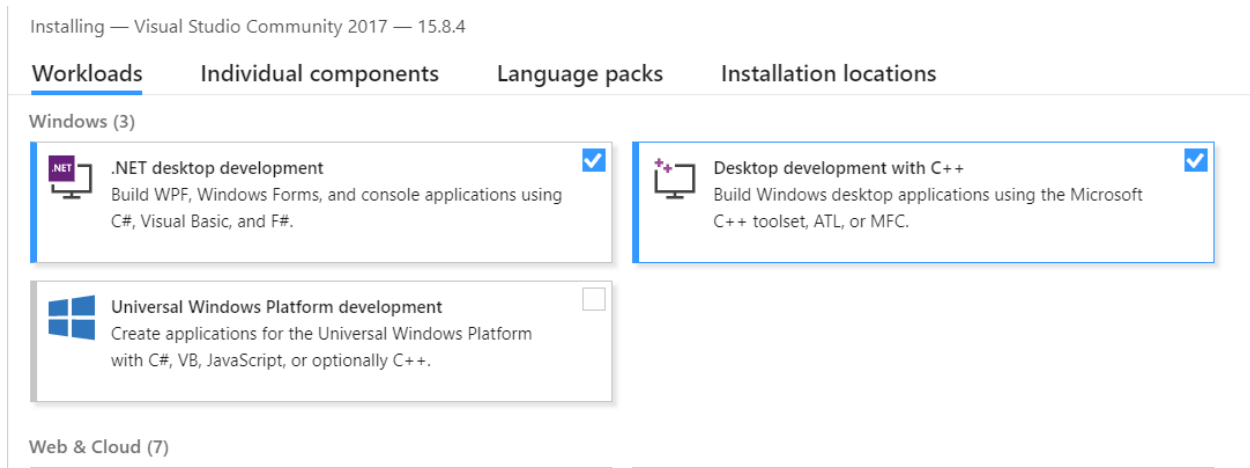


## Install Visual Studio 2017 Community

- Check the Desktop development with C++ in the install options.



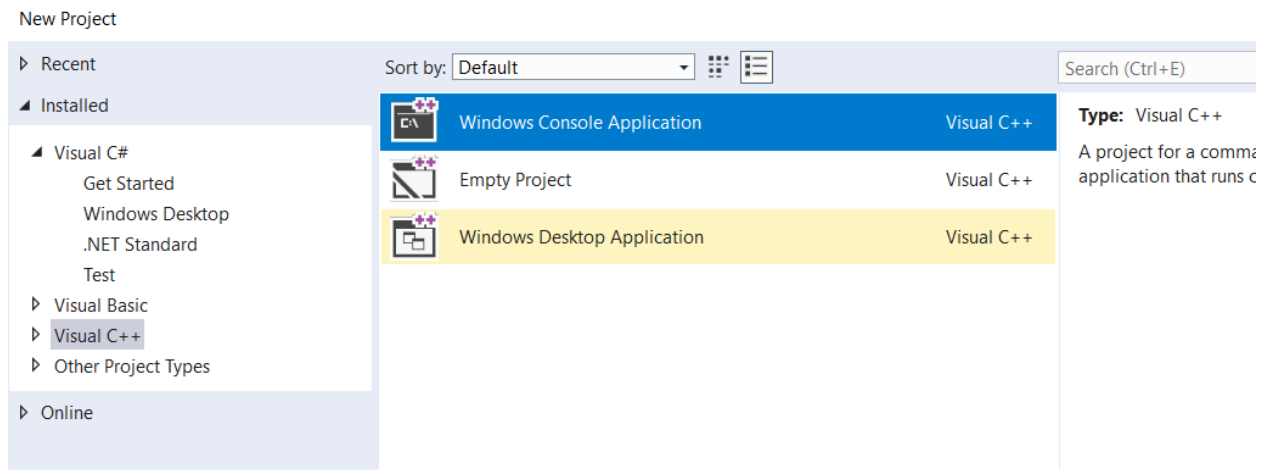
You may want to take 15 minutes to try the Hello World C++ tutorial:

<https://tutorials.visualstudio.com/cpp-console/intro>

or look it over as the install happens.

- 1) Start Visual Studio. You can sign in with your Clark account information. It will ask you to fill out some information – you can skip the optional Visual Studio Team Service Site.
- 2) Note that C++ compiler may not be installed with default options. You may be able to select it with the advanced install.
- 3) The first time you open Visual Studio, you can set the default configuration for General Use.

- 4) Go to File → New Project... and select Visual C++. You should see the options below, starting with “Windows Console Application”.



### Download and Extract OpenCV 3

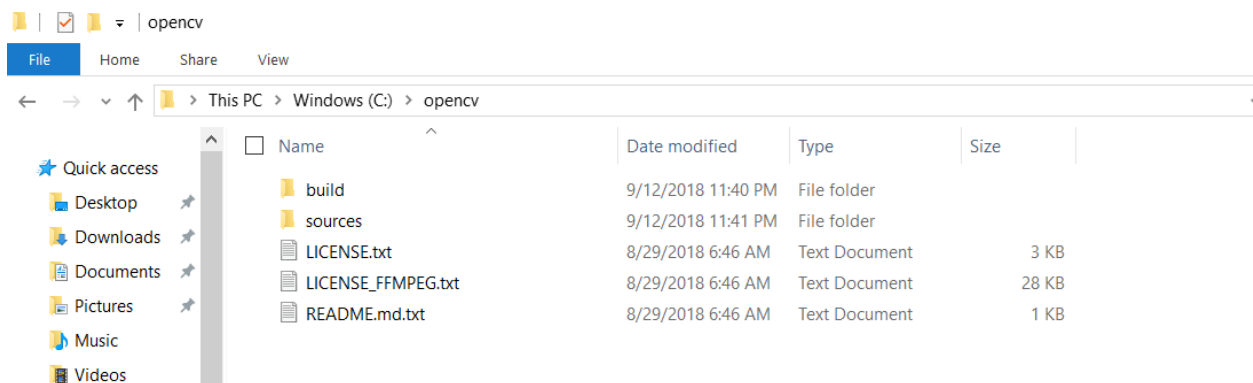
Download from <http://www.opencv.org>

Run the self-extracting executable.

Select where you would like to store the library.

On my computer, I extracted the library to c:\opencv\

You should see the build and sources subdirectories there:



### Set Environment Variables

You can set environment variables two ways.

One way is the *System* control panel. Open the *System* control panel, then select *Advanced system settings*. Then press the button for *Environment Variables...*



## Test the Path and Environment Variables

- 1) Open a new command prompt. (Press the Windows Key and then type *cmd*). If a command prompt is already open, you will need to exit and open a new one to refresh the environment variables.
- 2) Type ***opencv\_version***
  - a. If this outputs 3.4.3, the path is set properly.
  - b. If this outputs an error message, your path variable is not properly set.
- 3) Type ***cd %OPENCV\_DIR%***
  - a. You should current working directory should now be the location of the OpenCV's build\x64\vc15 directory.
  - b. If you type *dir*, the directory listing should have a bin and lib subdirectories.

## Configure Visual Studio

Start Visual Studio. If it was already open, you may need to exit and restart for the environment variables to take effect.

Creating a New Project:

- Create a new project (File → New → Project...)
- Select Visual C++ → Windows Console Application
- Type a name for the project (Default ConsoleApplication1 is ok for testing)
- Select a location – this is where the project will be stored.
  - Be careful on lab computers, the default location is on the C: drive
- Press OK.

Configure the Project:

- 1) Select Build → Configuration Manager...
- 2) For each project, Under Platform, select x64
- 3) For Active Solution Platform, select x64

## Configuring the Library

There are two ways to configure the library:

- On a per-project basis. The settings would need to be configured for every project. To configure the project, make sure the current project is selected in the Solution Explorer (top-right sub-window) select:

Project → Properties

Alternatively: right-click on the project and select Properties

Make sure Configuration says Active(Debug) and Platform says Active(x64).

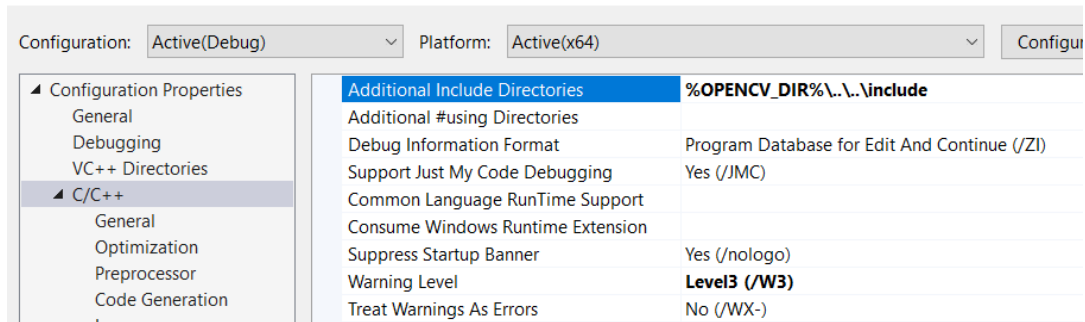
- On a per-user basis. [This seems to have changed in Visual Studio 2017. There's probably still a way to do this, but it's not clear how.]

**There are four settings that have to be made in either case:**

1. Tell the compiler where the include (.hpp) files are located
2. Tell the linker where the library (.lib) files are located
3. Tell the linker what library (.lib) files to link
4. Tell the debugger the path to the *bin* folder (to find the .dll files)

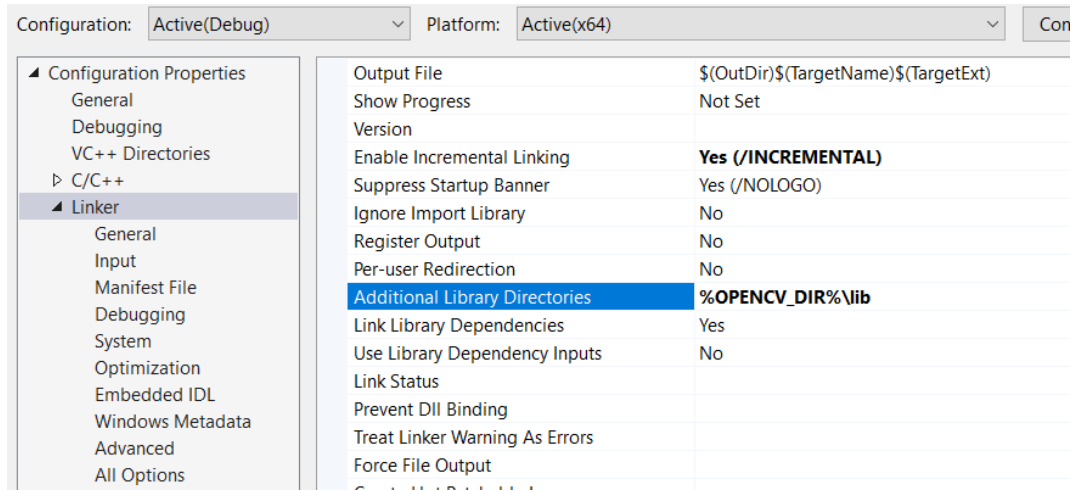
## 1) Select C/C++

- Set Additional Include Directories: %OPENCV\_DIR%\..\..\include



## 2) Select Linker

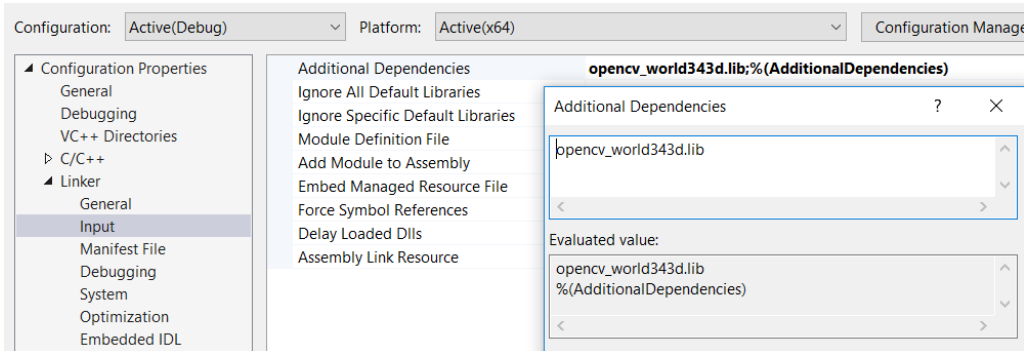
- Set Additional Library Directories: %OPENCV\_DIR%\lib



### 3) Select Linker → Input

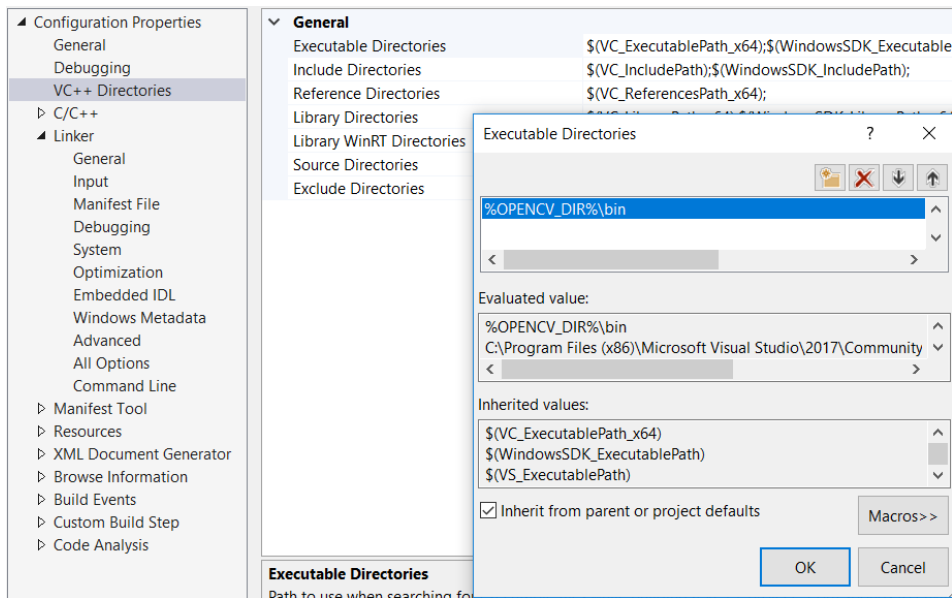
- a. Edit the *Additional Dependencies* to add *opencv\_world343d.lib*

Note: If you are using a different version of OpenCV than 3.4.3, check your lib directory for the appropriate name. The “d” at the end means “debug” version.



### 4) Select VC++ Directories

- a. Edit the *Executable Directories* to add *%OPENCV\_DIR%\bin*



## Your First OpenCV Program

Edit the .cpp file, replacing the existing starter code with the following:

```
#include "pch.h"           // remove this if not using precompiled headers
#include <opencv2/opencv.hpp> //Include file for every supported OpenCV function
int main(int argc, char** argv) {
    cv::Mat img = cv::imread(argv[1], -1);
    if (img.empty()) return -1;
    cv::namedWindow("Example1", cv::WINDOW_AUTOSIZE);
    cv::imshow("Example1", img);
    cv::waitKey(0);
    cv::destroyWindow("Example1");
    return 0;
}
```

This code reads an image and displays it. Then exits when a key is pressed.

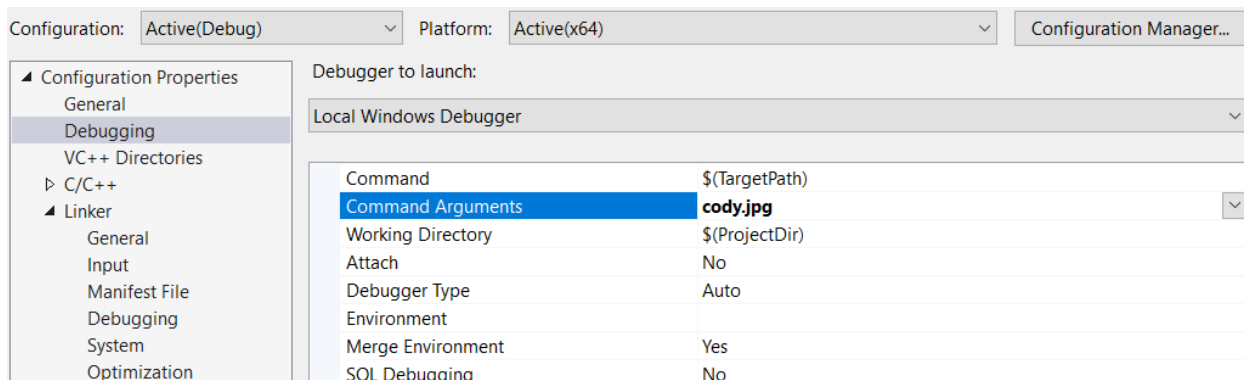
Find a small image on the internet or your computer. Copy the image into the project's directory (the same folder as your .cpp file). On my computer, this directory was

projects\ConsoleApplication1\ConsoleApplication1\

Note that the first ConsoleApplication1 is the “solution” directory, and the second one is the “project” directory. That will be the “current working directory” when we run our programs.

The main method accepts a command-line argument. To set this argument in the debugger, edit the Project Properties.

- Select *Debugging*
  - Add the file name to the *Command Arguments* option.




## Build and Test

Select Build → Build Solution

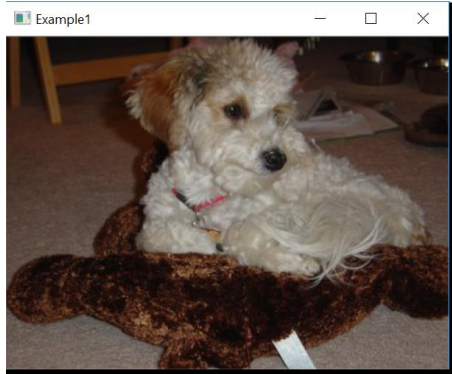
If everything is configured correctly, it should compile and tell you that:

===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====



To run: Debug → Start Debugging (or )

You should see a window pop-up with your image!



**Congratulations! You're ready to explore OpenCV!**