GATS Companion to Installing BOOST

Author: Garth Santor
Editors: Trinh Hān

Version: 1.0.0 (2020-09-09)

Overview

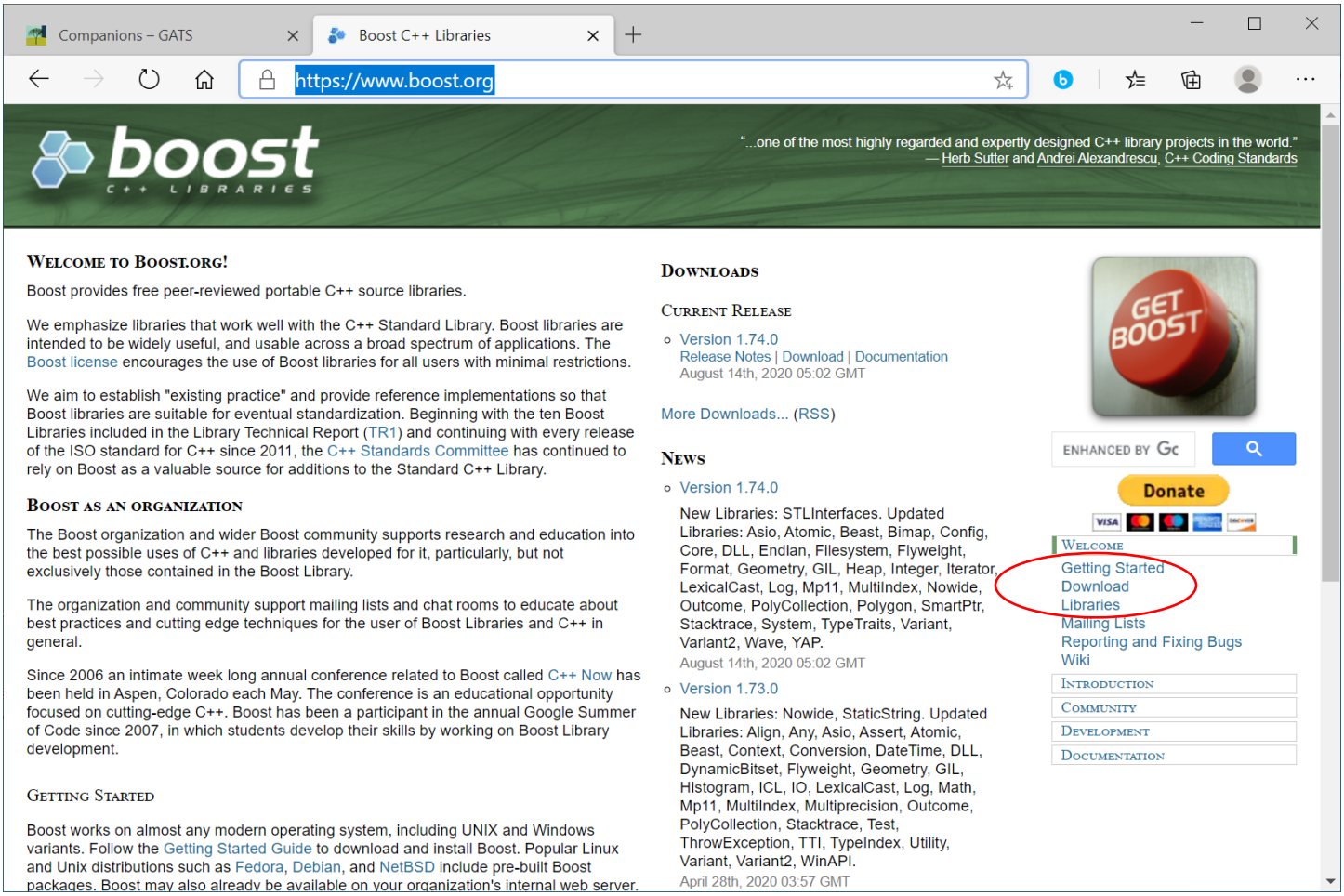
How to install BOOST for use within Visual Studio 2019.

boost version: 1.74.0

Visual Studio version: 16.7.4

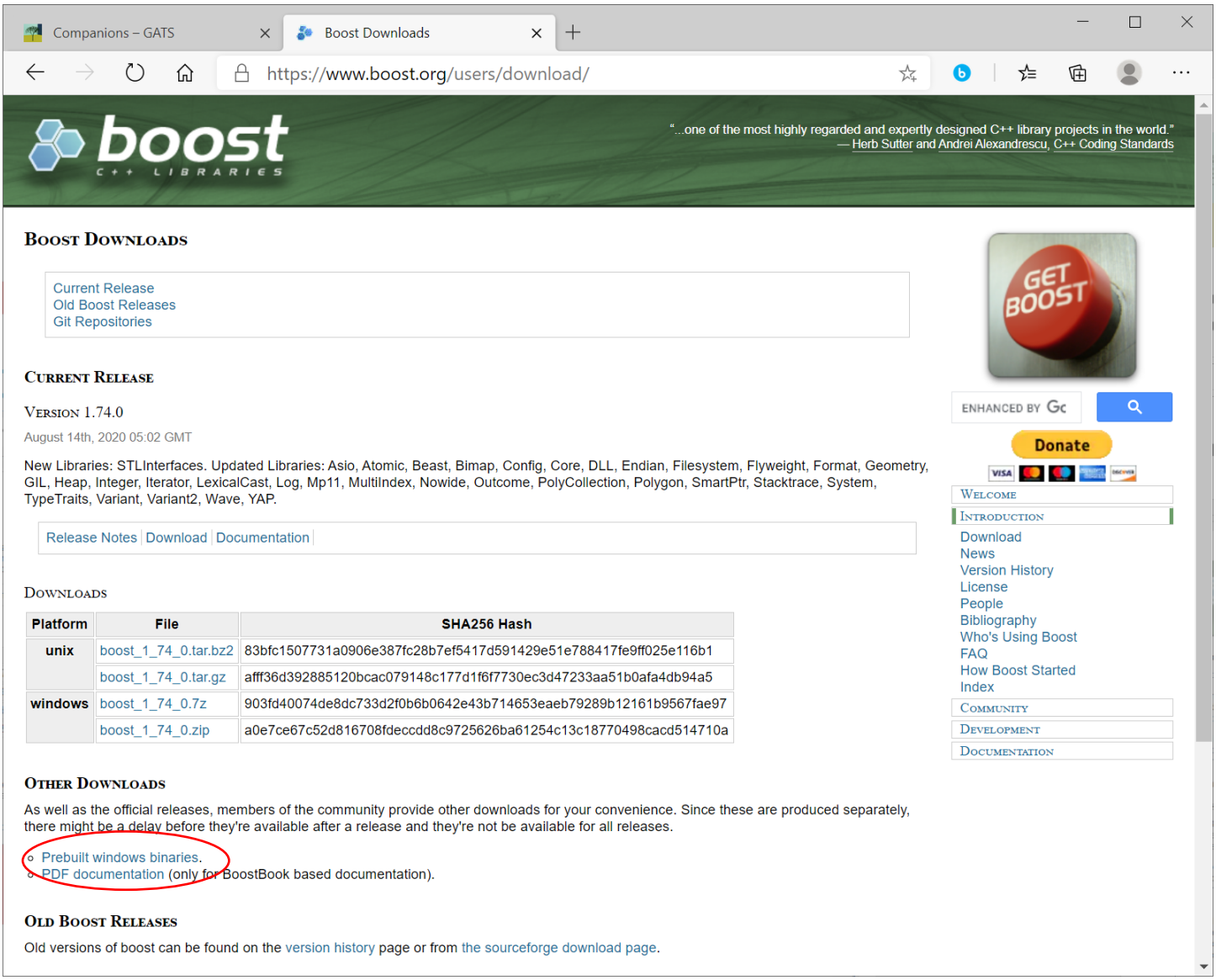
Acquiring the files

Go to the website https://www.boost.org/ and click the **download** link on the right side of the window.



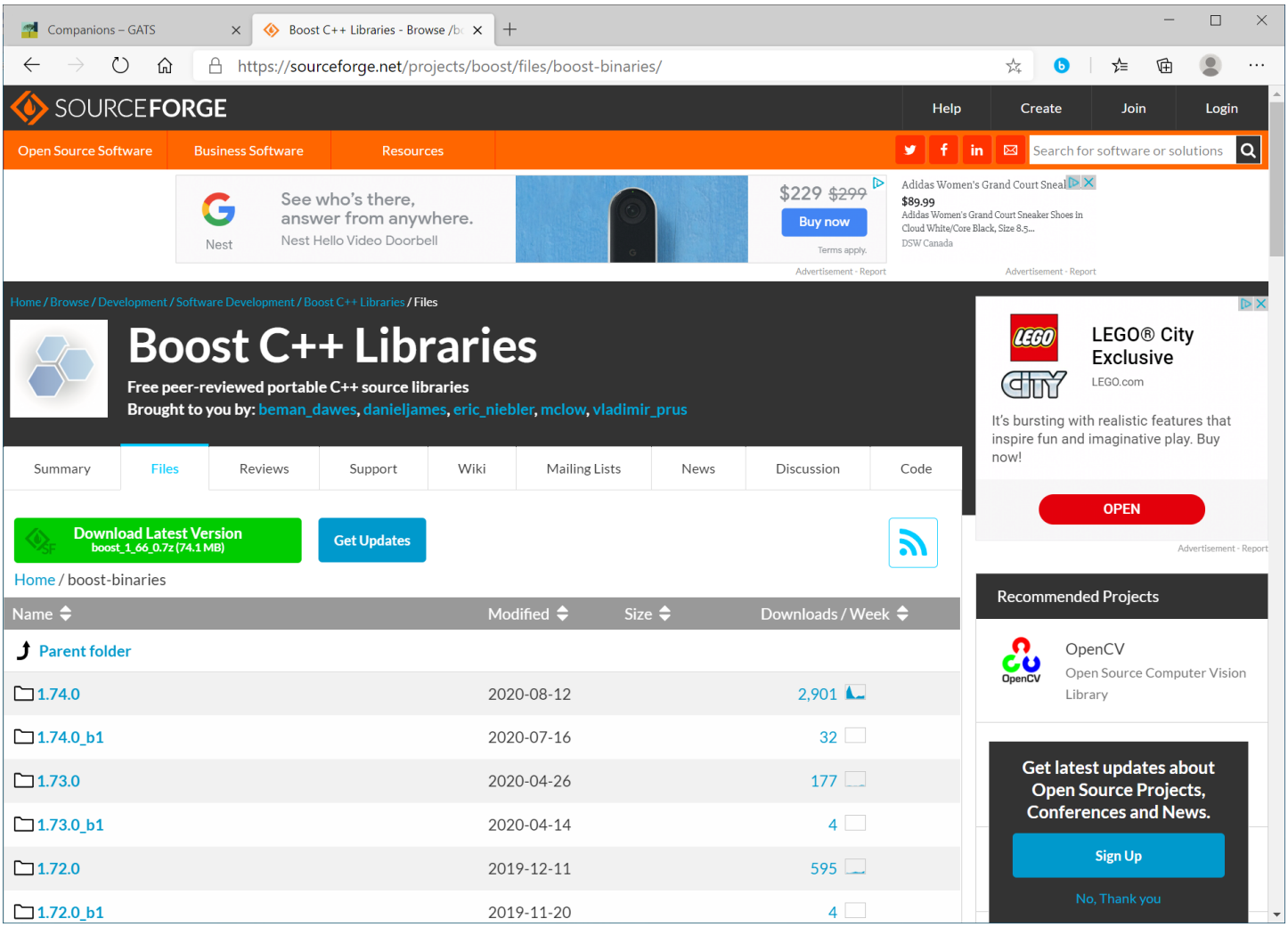
Pick the *prebuilt windows binaries* link.

https://www.boost.org/users/download/



You'll now be on SourceForge. Pick the latest *1.74.0* version link.

https://sourceforge.net/projects/boost/files/boost-binaries/

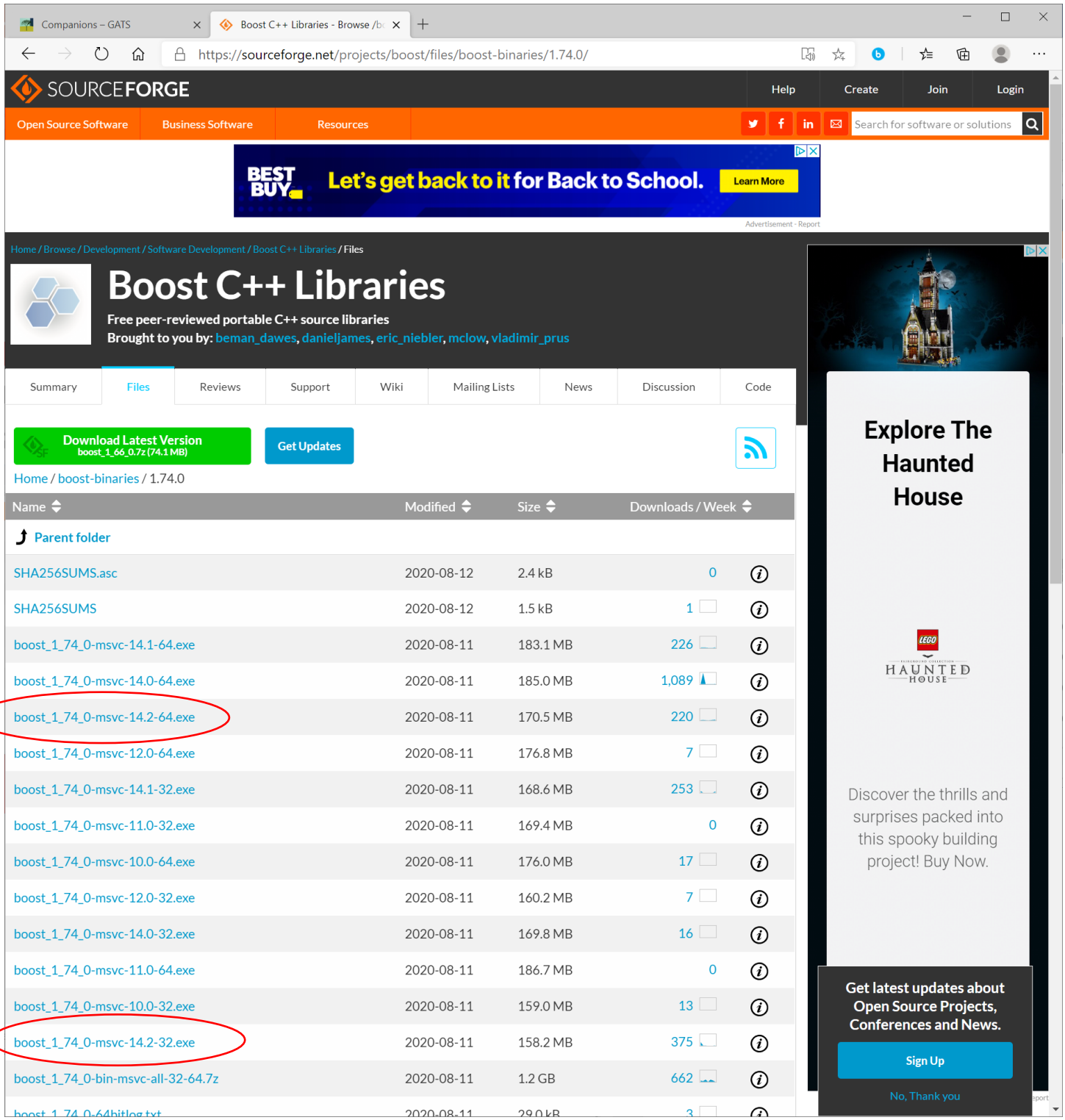


There are two binaries you'll need to install, a 32-bit version, and a 64-bit version. You'll also have to select the release for the current version of Visual Studio 2019. The currently updated version of Visual Studio 2019 is compiler is 16.7.3, but the binary format is 14.2, so that's the version we will use.

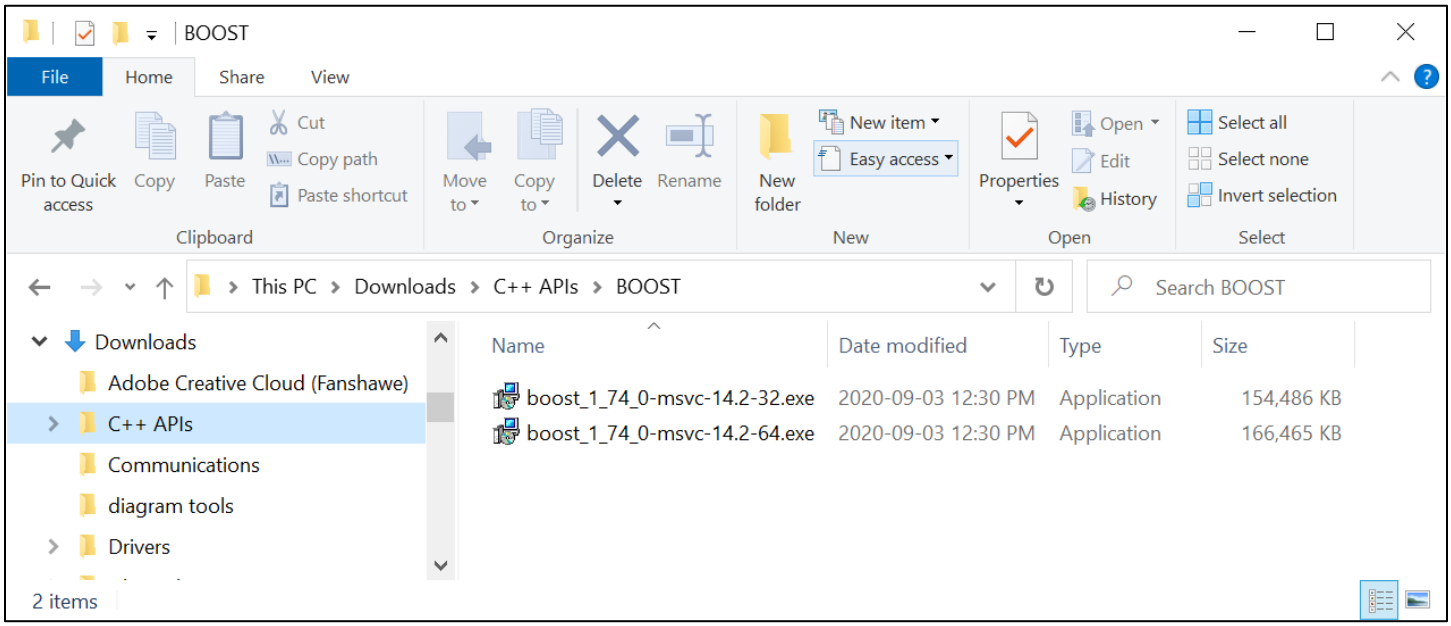
https://sourceforge.net/projects/boost/files/boost-binaries/1.74.0/

Download both:

- boost_1_74_0-msvc_14.2-32.exe
- boost_1_74_0-msvc_14.2-64.exe


Companions – GATS
Boost C++ Libraries - Browse /b
https://sourceforge.net/projects/boost/files/boost-binaries/1.74.0/
SOURCEFORGE
Help
Create
Join
Login
Open Source Software
Business Software
Resources
Search for software or solutions
BEST BUY
Let's get back to it for Back to School.
Learn More
Advertisement - Report
Home / Browse / Development / Software Development / Boost C++ Libraries / Files
Boost C++ Libraries
Free peer-reviewed portable C++ source libraries
Brought to you by: beman_dawes, danieljames, eric_niebler, mclow, vladimir_prus
Summary
Files
Reviews
Support
Wiki
Mailing Lists
News
Discussion
Code
Download Latest Version
boost_1_66_0.7z (74.1 MB)
Get Updates
Home / boost-binaries / 1.74.0
Name
Modified
Size
Downloads / Week
Parent folder
SHA256SUMS.asc 2020-08-12 2.4 kB 0
SHA256SUMS 2020-08-12 1.5 kB 1
boost_1_74_0-msvc-14.1-64.exe 2020-08-11 183.1 MB 226
boost_1_74_0-msvc-14.0-64.exe 2020-08-11 185.0 MB 1,089
boost_1_74_0-msvc-14.2-64.exe 2020-08-11 170.5 MB 220
boost_1_74_0-msvc-12.0-64.exe 2020-08-11 176.8 MB 7
boost_1_74_0-msvc-14.1-32.exe 2020-08-11 168.6 MB 253
boost_1_74_0-msvc-11.0-32.exe 2020-08-11 169.4 MB 0
boost_1_74_0-msvc-10.0-64.exe 2020-08-11 176.0 MB 17
boost_1_74_0-msvc-12.0-32.exe 2020-08-11 160.2 MB 7
boost_1_74_0-msvc-14.0-32.exe 2020-08-11 169.8 MB 16
boost_1_74_0-msvc-11.0-64.exe 2020-08-11 186.7 MB 0
boost_1_74_0-msvc-10.0-32.exe 2020-08-11 159.0 MB 13
boost_1_74_0-msvc-14.2-32.exe 2020-08-11 158.2 MB 375
boost_1_74_0-bin-msvc-all-32-64.7z 2020-08-11 1.2 GB 662
Explore The Haunted House
LEGO
HAUNTED HOUSE
Discover the thrills and surprises packed into this spooky building project! Buy Now.
Get latest updates about Open Source Projects, Conferences and News.
Sign Up
No, Thank you

You should now have the two new files in your download folder…



Install the libraries

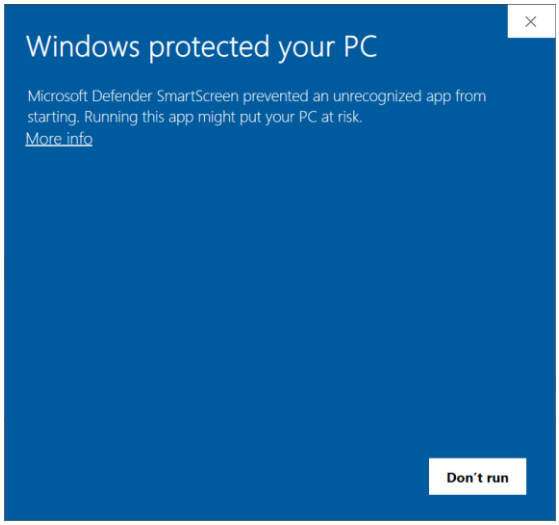
The two files you downloaded are self-extracting archives. You'll need about 4GiB of free memory (ideally on drive C). There are two reasons drive C is the best choice:

Compatibility: System configuration and portability will be better (hopefully seamless) if we all use the same location.

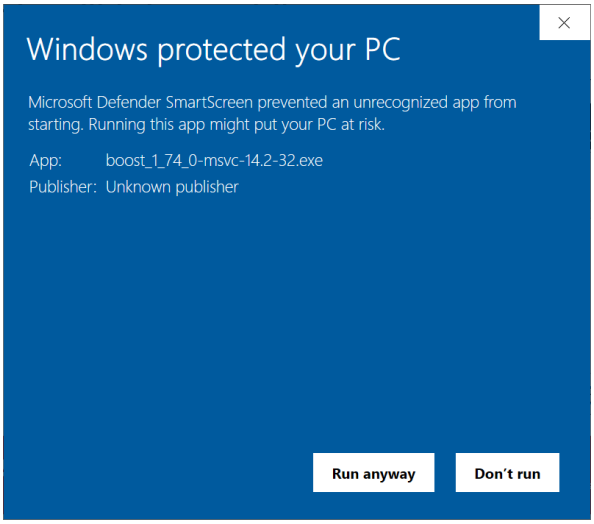
Performance: Many of us will have a solid-state drive (SSD) as their primary drive. I have a ½ TiB SSD for my primary drive (drive C:) and a 2 TiB HDD for my secondary drive (drive D:). Since boost has a huge number of small header files (14,184 in version 1.73.0) the lower latency and higher transfer rate of an SSD will provide a noticeable improvement when compiling.

You'll install each library (one after the other) as many of the files will be overwritten by the other. The order you install them doesn't matter. Double click either archive and follow the steps below.

You'll probably see this as soon as you run the installer.

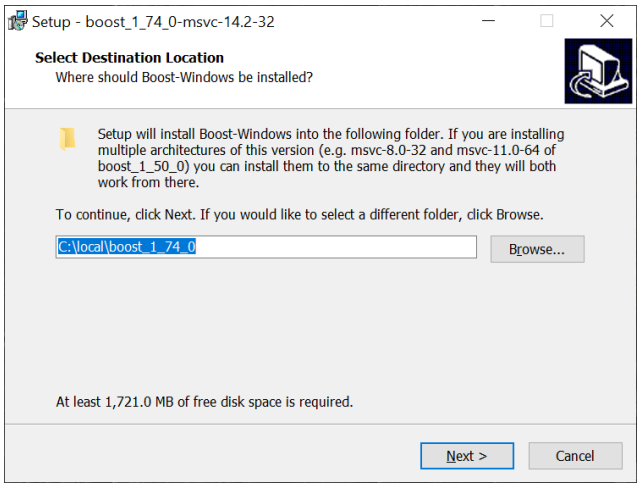


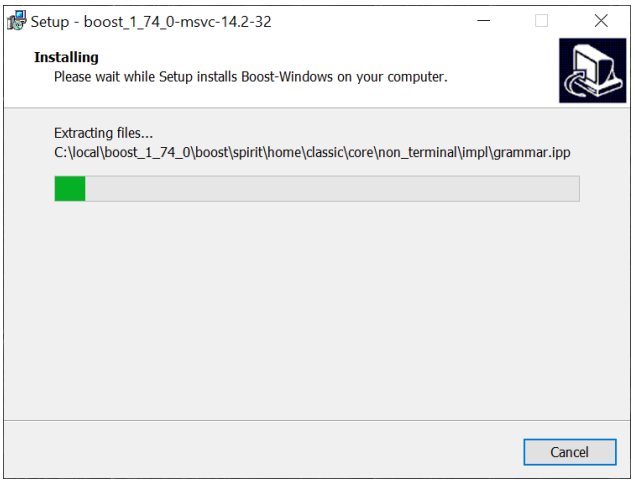
Click More info, you should see:



Click **Run anyway**.

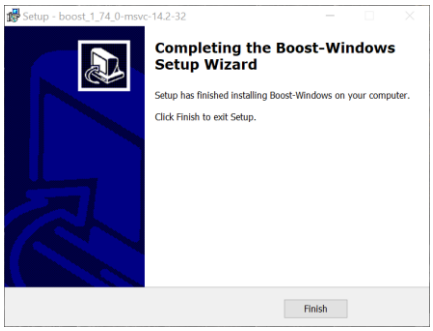
It is best to use the default folder (C: drive), so just click **Next >**.

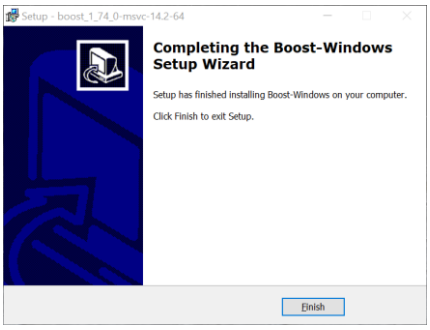




Installation could take several minutes, depending on the speed of your drive, RAM, and CPU. My desktop installs boost in about 1 minute, but I've seen some system take more than 20 minutes to install.

When complete, do the same for the 64-bit version.



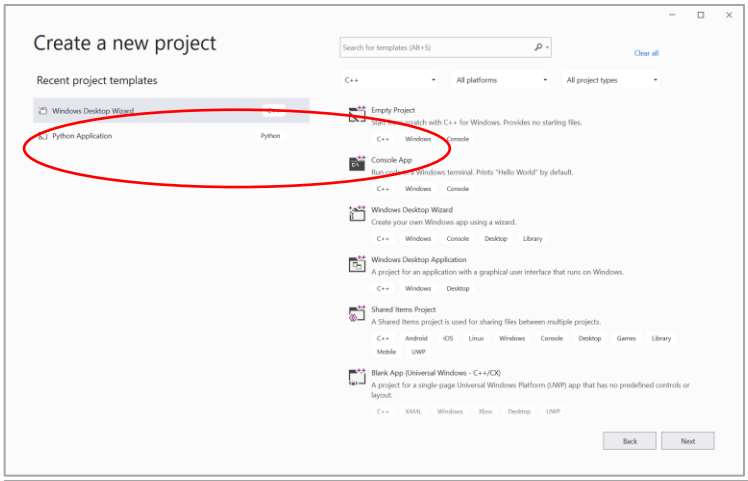


Creating Property Pages for Visual C++

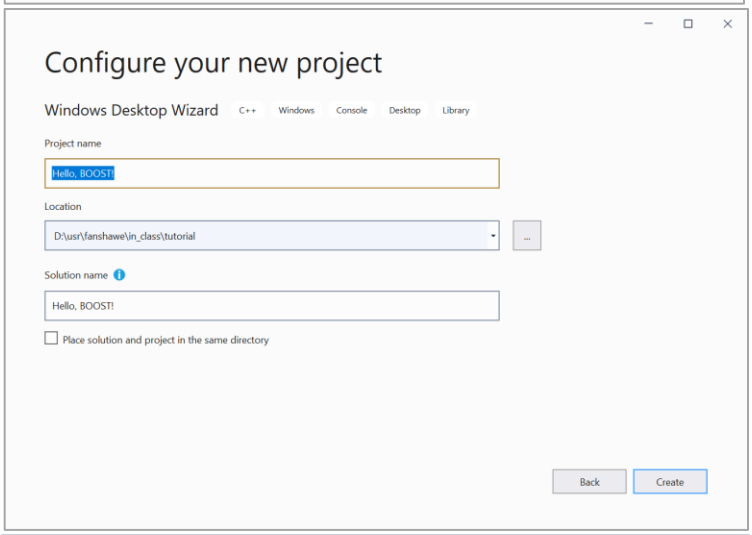
Visual Studio 2017 and earlier had global property pages, but they were removed in Visual Studio 2019 (they haven't said why…)

We'll need a new C++ project loaded into Visual Studio 2019. We'll create a *Hello, BOOST!* application to test our configuration.

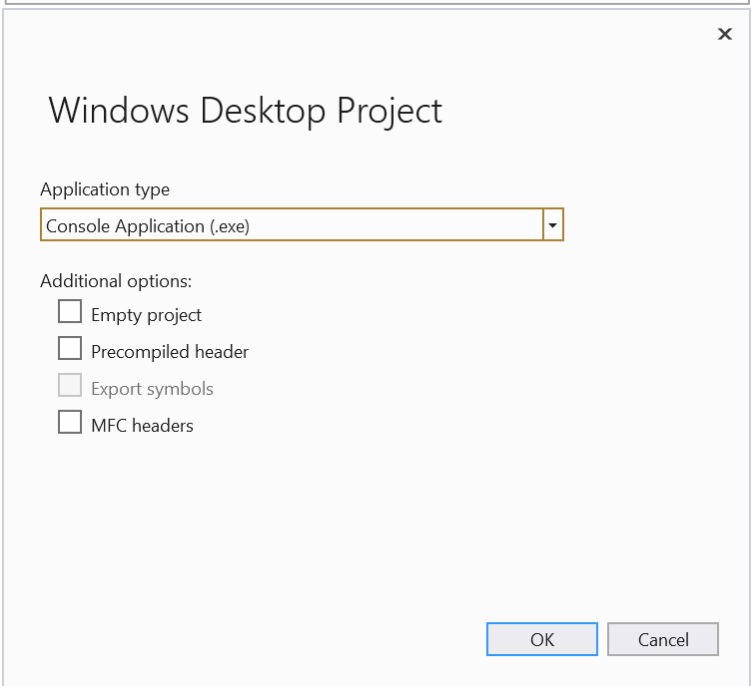
Use the **Windows Desktop Wizard** to create a hello world desktop application.



Name the project…



Verify **Console Application (.exe)** as the application type, no additional options need to be checked.



Ignoring the comments your source code should appear as:

```
#include <iostream>

int main() {
    std::cout << "Hello World!\n";
}
```

Modify the code to use a *header-only* boost library.

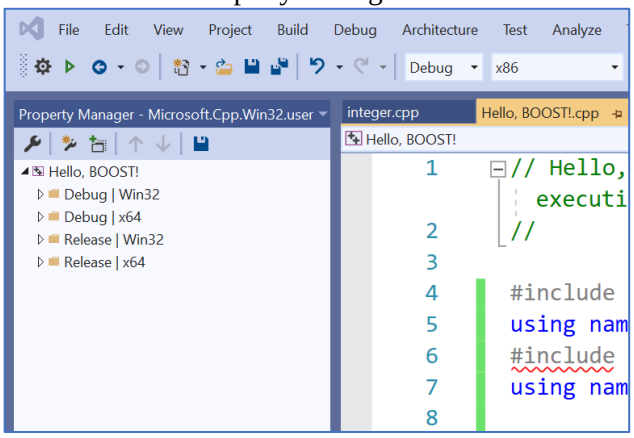
```
#include <iostream>
using namespace std;
#include <boost/multiprecision/cpp_int.hpp>
using namespace boost::multiprecision;

int main() {
    cout << "Hello World!\n";
    cpp_int big("12345678901234567890123456789012345678901234567890");
}
```

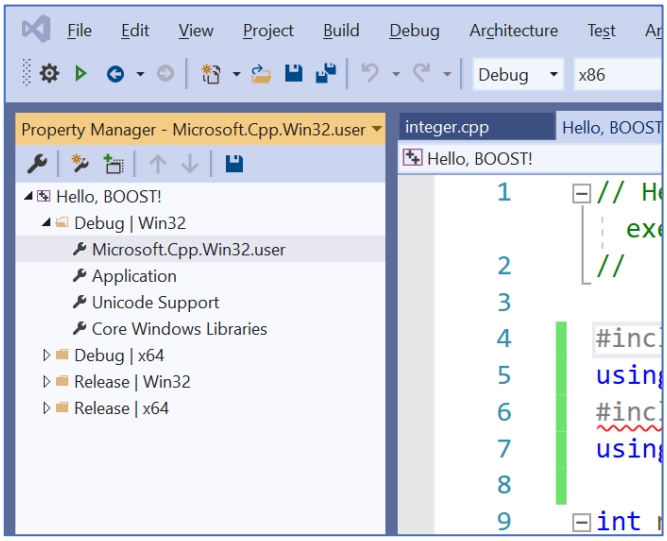
Add a new *reusable* project property sheet (32-bit)

Open the Property Manager

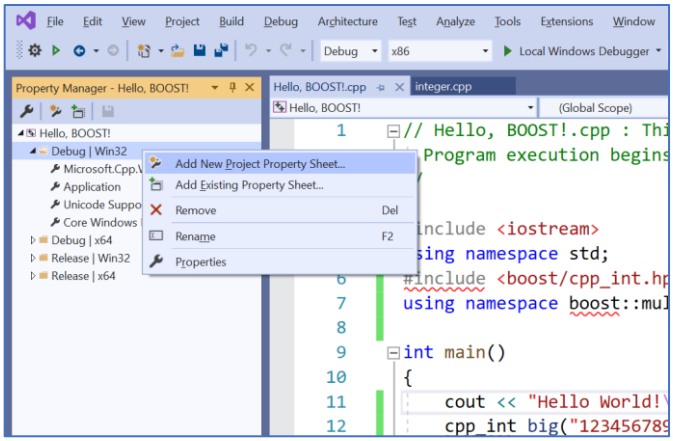
Menu: View → Property Manager



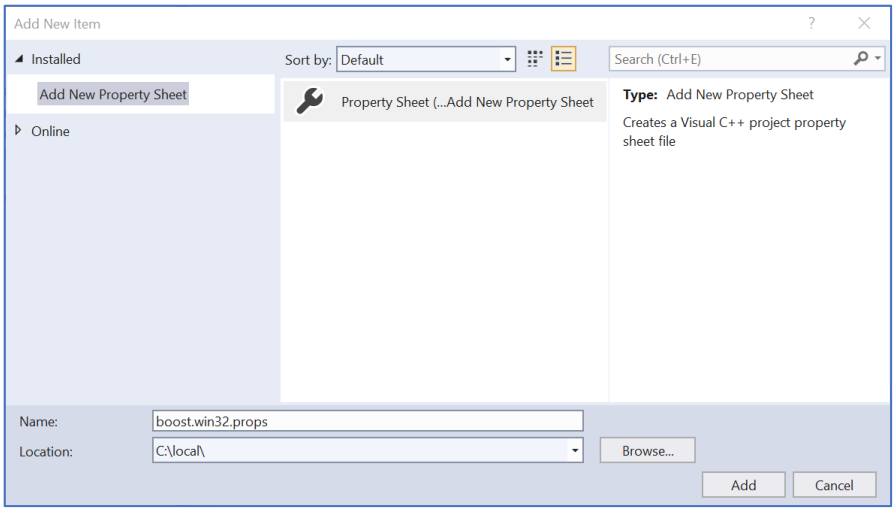
Expand the Debug | Win32 folder



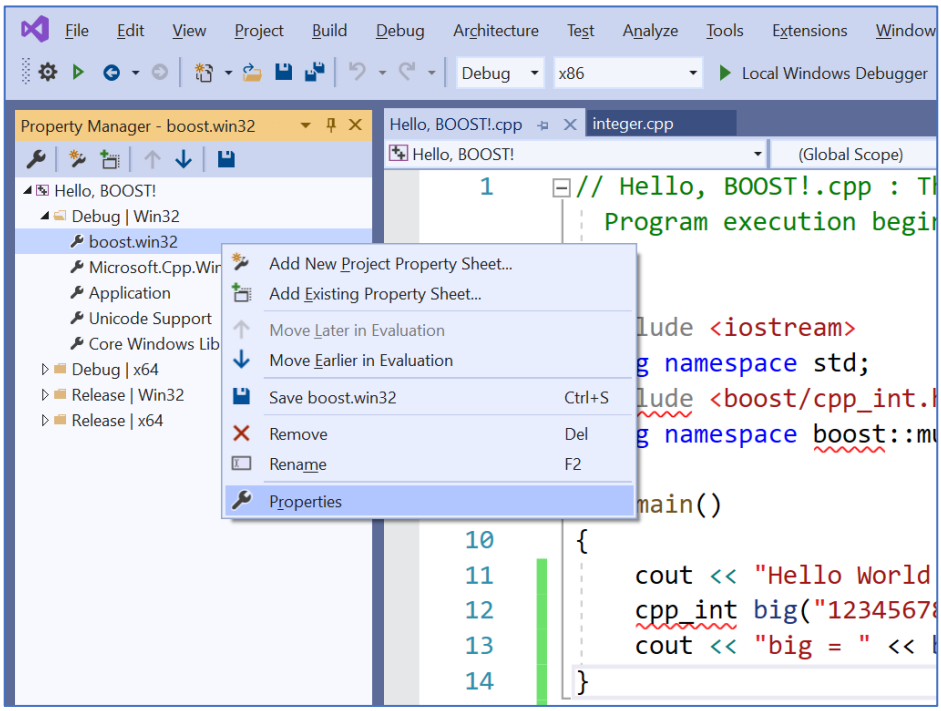
Add a new project property sheet



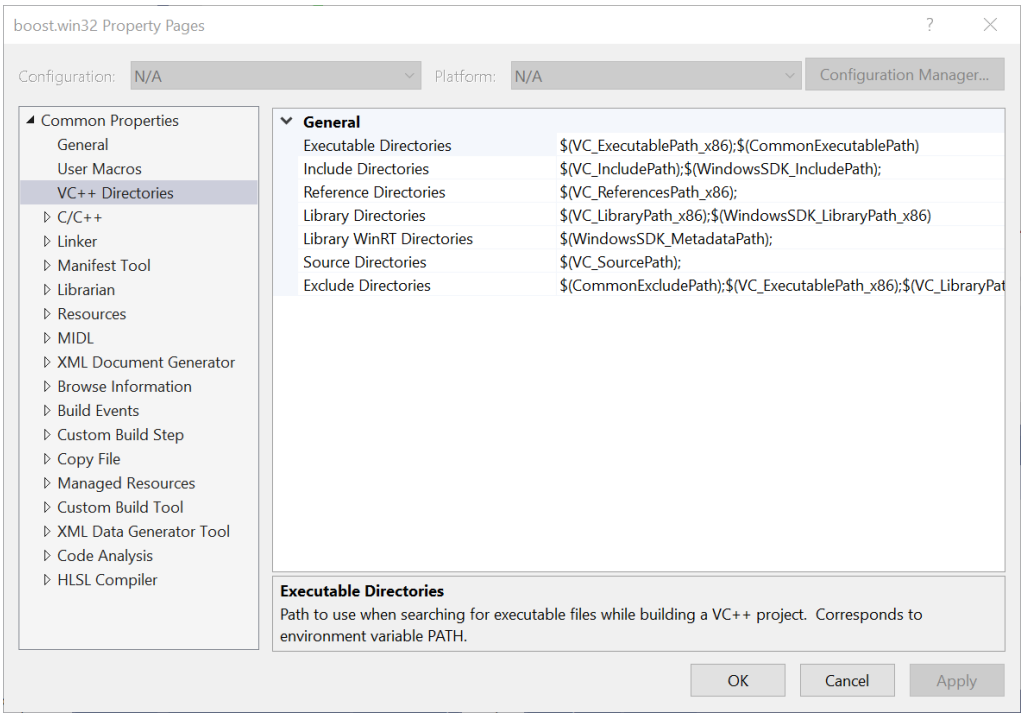
Name the file **boost.win32.props** and save the property sheet to the boost installation folder **C:\local**.



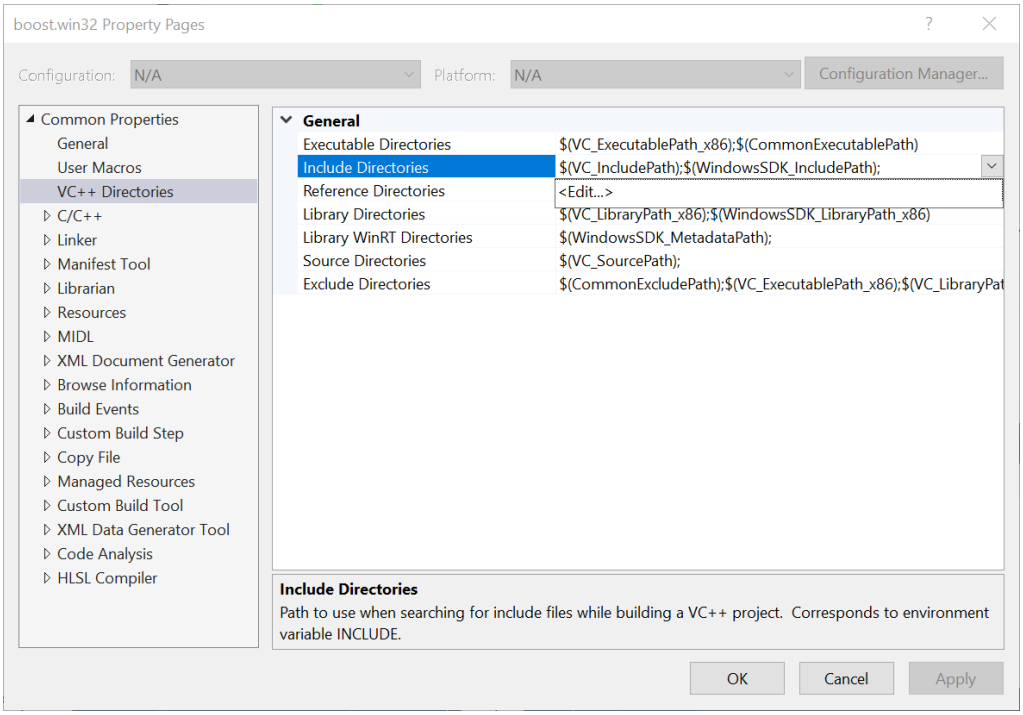
Open the property sheet.



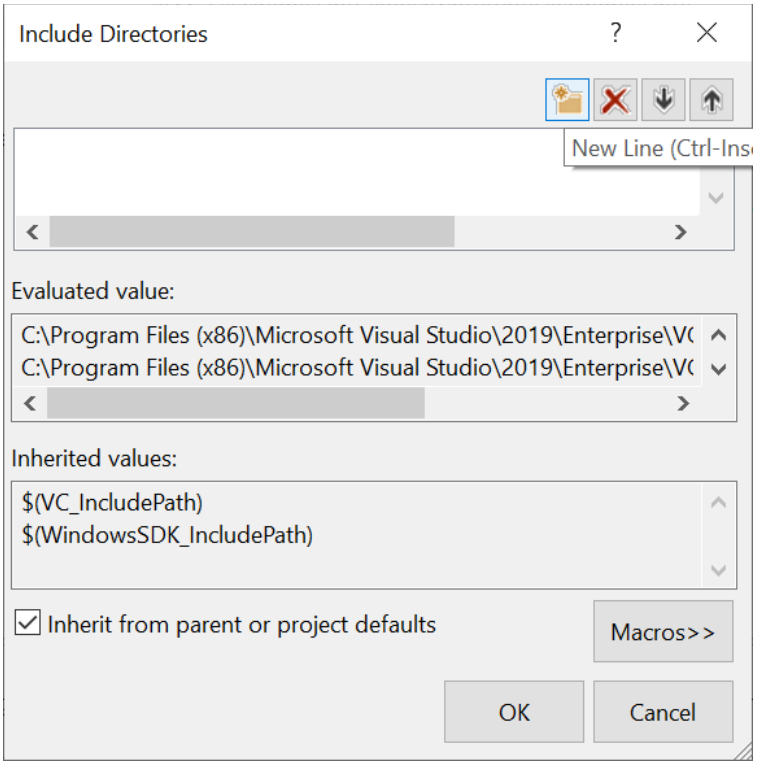
Select the **VC++ Directories** tab.



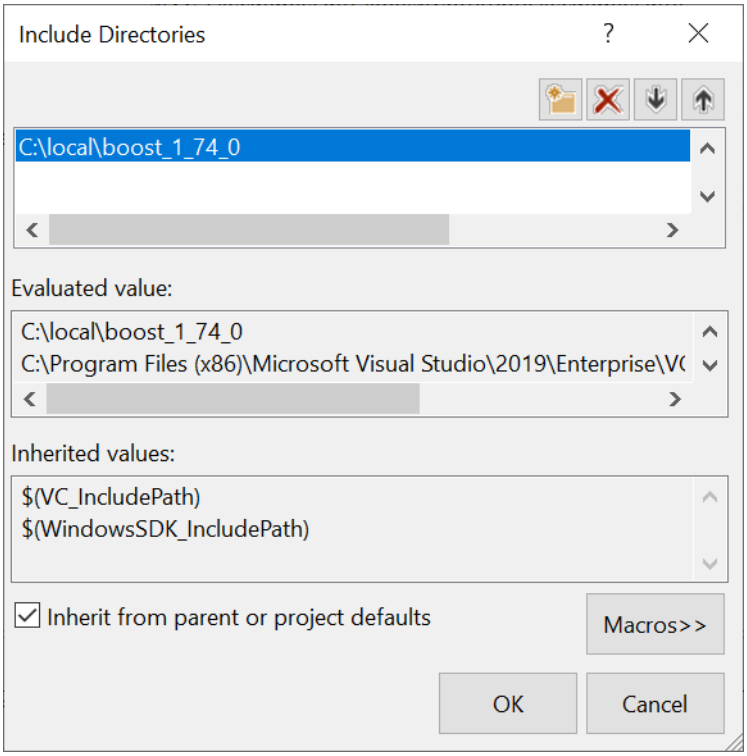
Edit the include paths.



Add a new line…

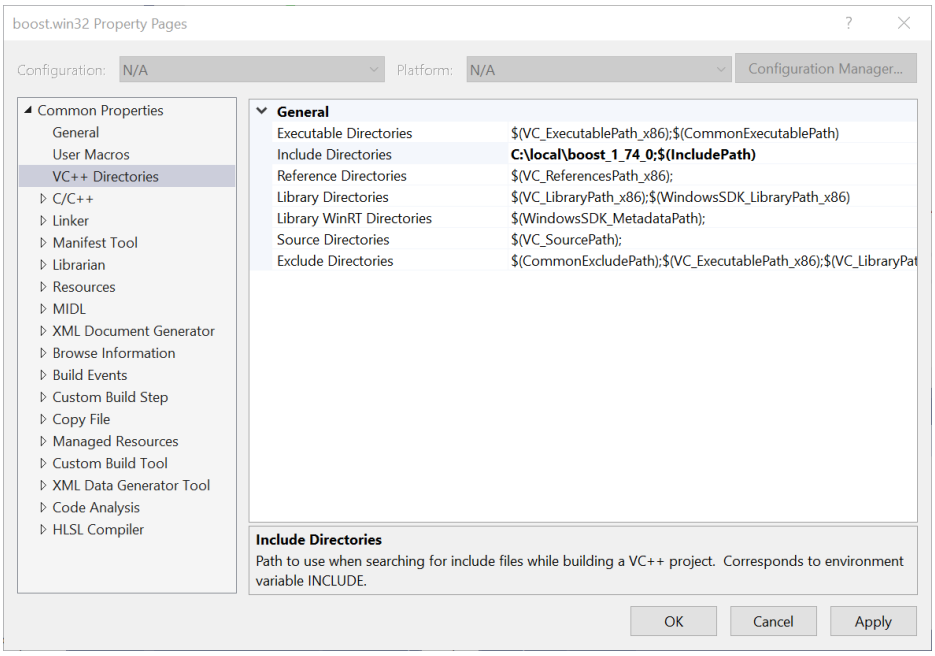


Add the boost main folder



You should see…

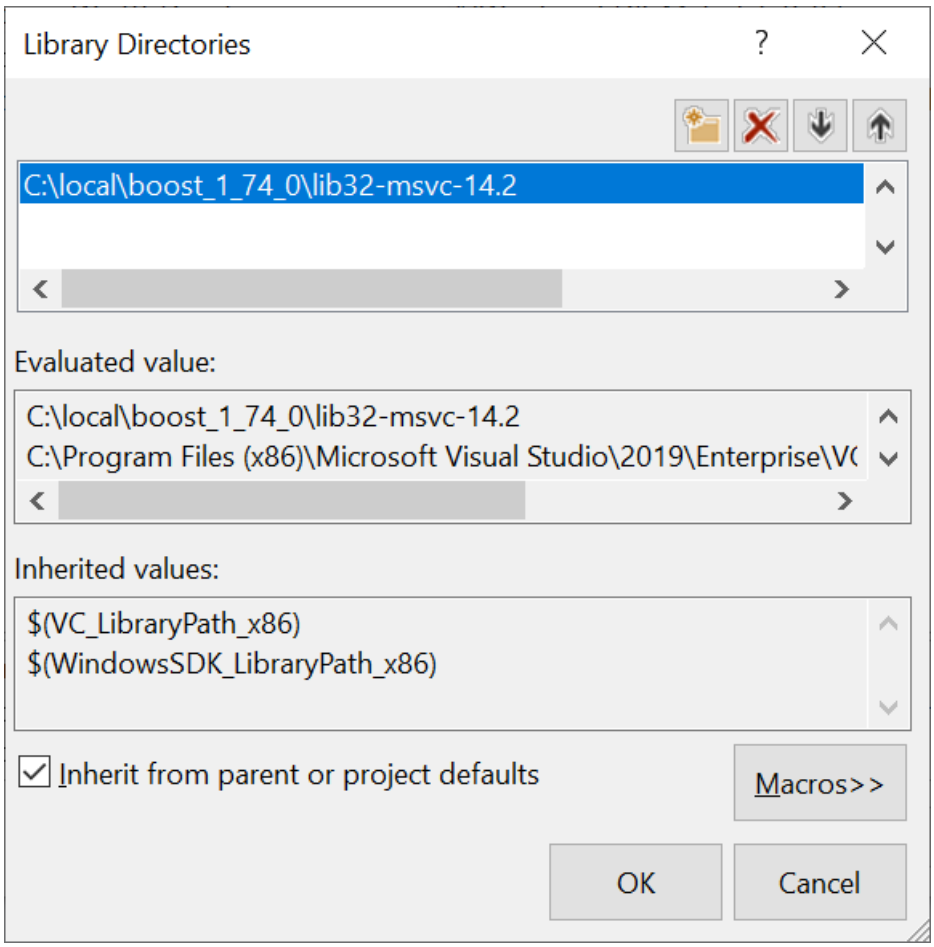
The $(IncludePath) must be there as it represents the original path from the default configuration.



Add to the library path.

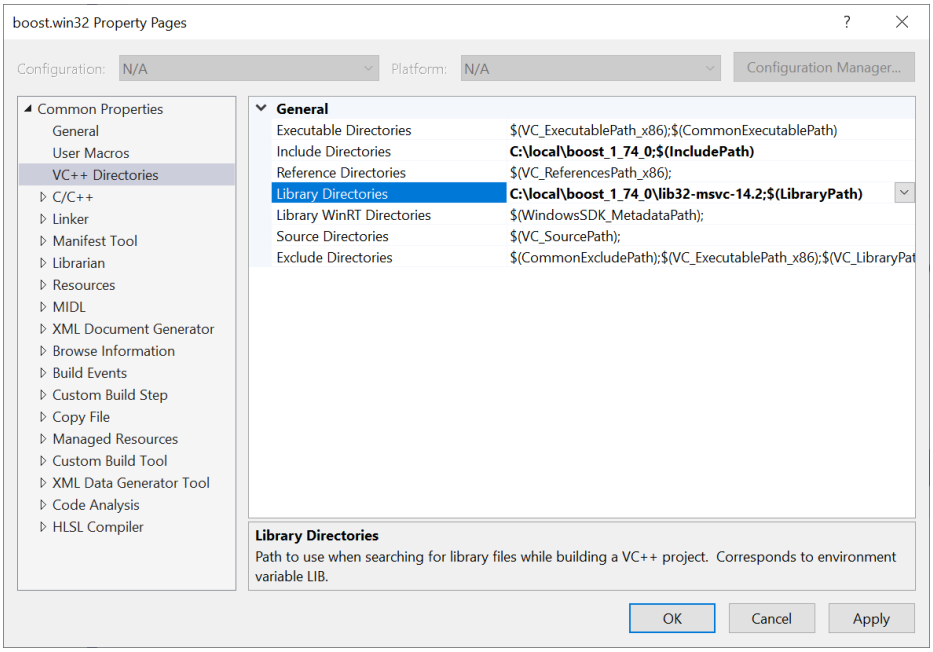
Note that we must specify which of the two libraries we want.

In this case we want the Win32 library, so we pick the **lib32-msvc-14.2** folder



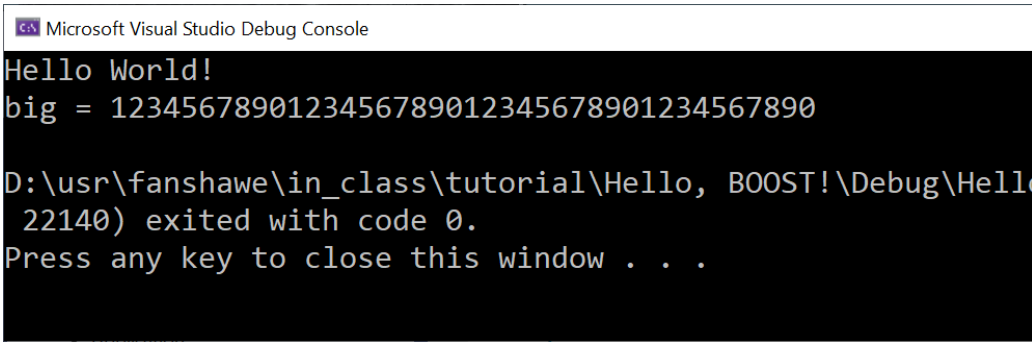
You should see…

Click OK



Test the configuration…

Run the program.
The errors should disappear and the project should build and execute.

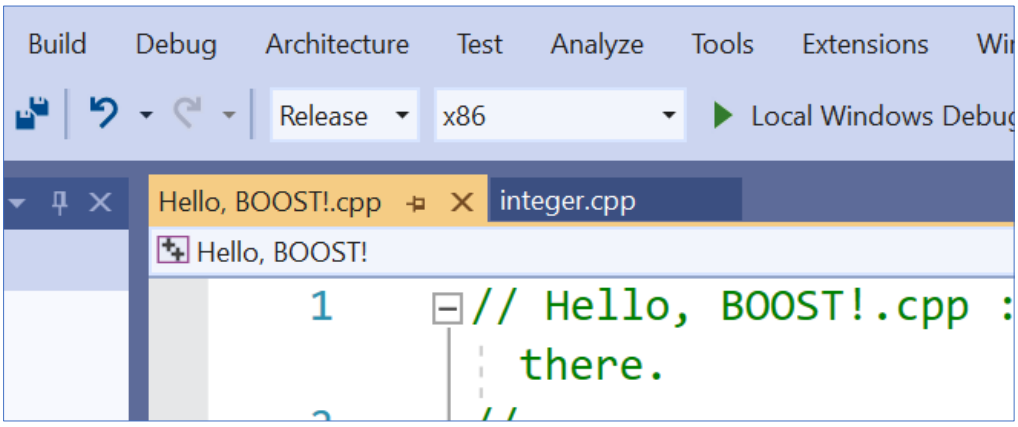


Adding the property sheet to release mode

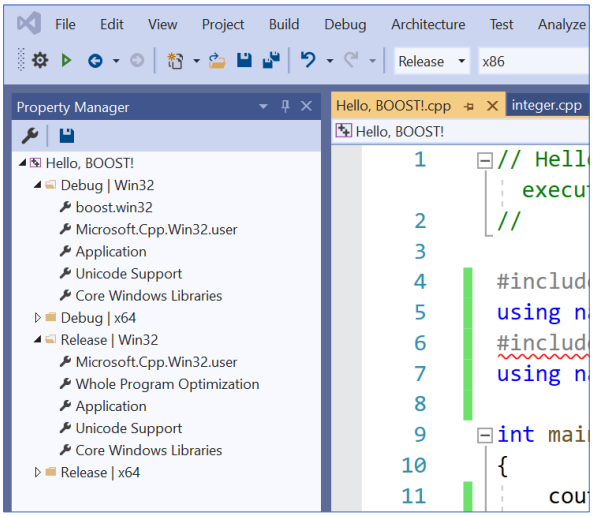
The release build uses the same header files as debug mode, and those header files instruct the linker to use the correct debug/release libraries. So, we can reuse the property page from the debug build in the release build.

Switch to release mode.

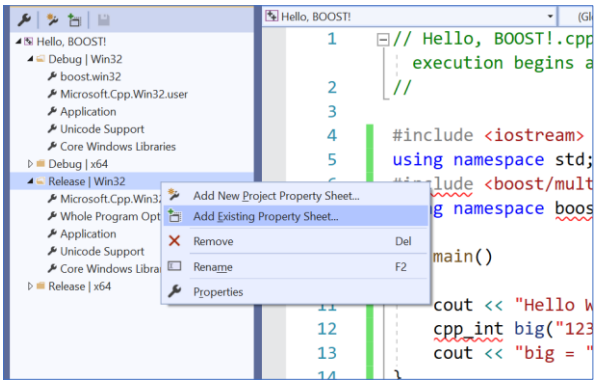
You'll notice that the include directive, the using statement, and the object definition are generating errors again.



Expand the Release | Win32 tab in the Property Manager

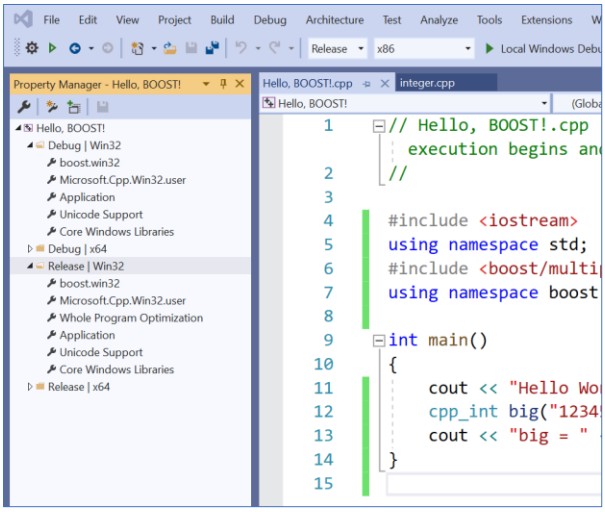


Add the existing property sheet.

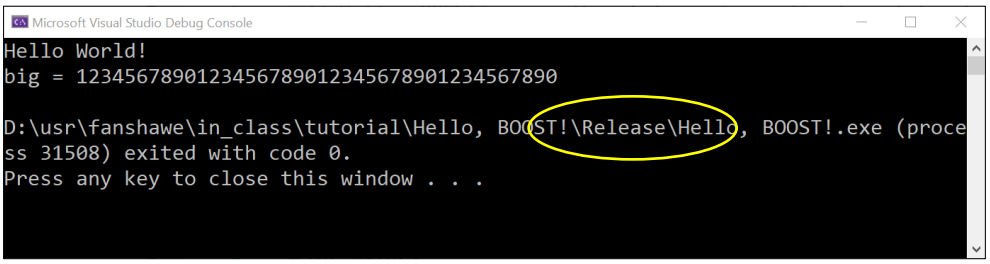


You should see…

And note that the errors have cleared and



Verify the release mode builds



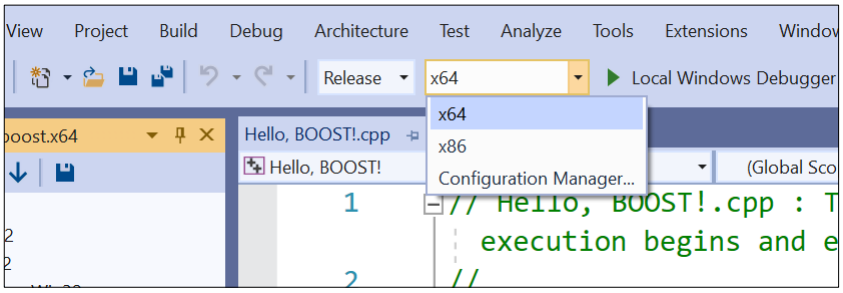
Configure for 64-bit builds

The 64-bit (x64) configurations use the same header files, but different libraries. Libraries specifically built with the 64-bit compiler.

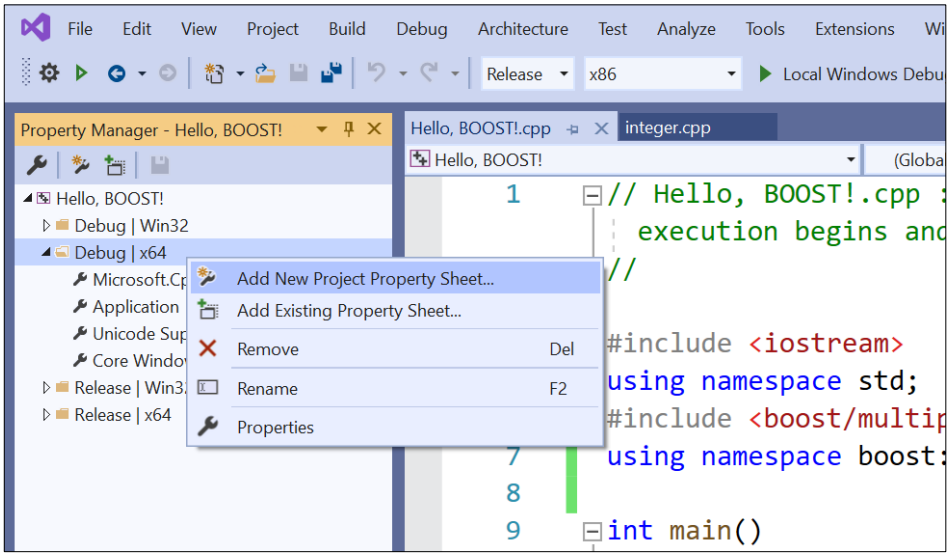
The process will be similar to the x86 installation:

- Create a property sheet in the boost folder
- Set the paths: same include, 64-bit binaries
- Add to both configurations.

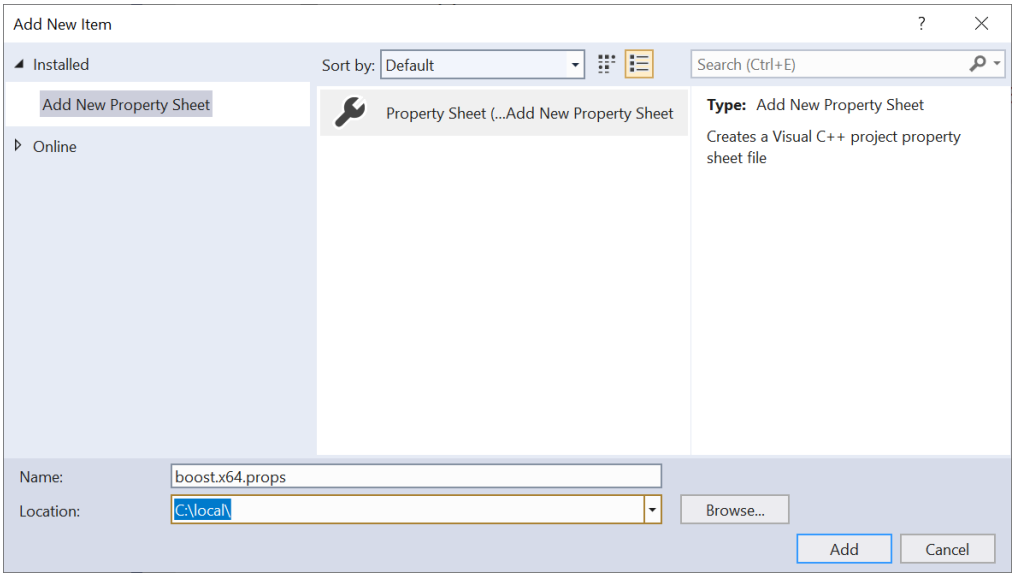
Switch to the 64-bit (x64) build.



Add a new project property sheet to Debug | x64

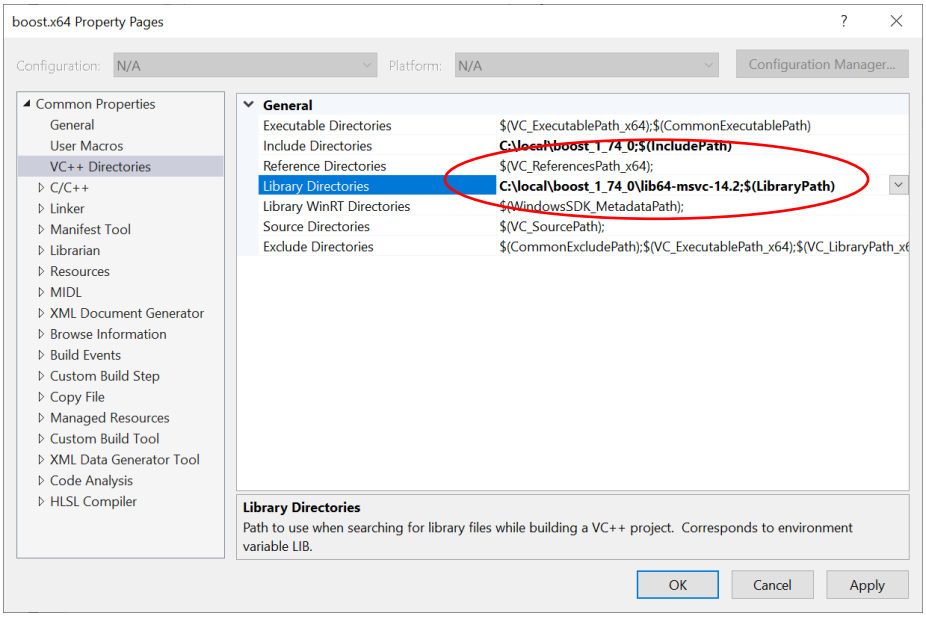


Name the file boost.x64.props and save it in the C:\local folder.



Configure the paths.

Note that the library path points to the 64-bit folder.



Add the file you just created to the release configuration as well.

Save and test both configurations

New projects

For new projects you'll only need to add the existing property sheets to the appropriate project type in the project manager.