GATS Companion to Installing BOOST

Author: Garth Santor
Editors: Trinh Hān
Copyright Dates: 2009, 2020, 2023
Version: 1.1.0 (2023-10-29)

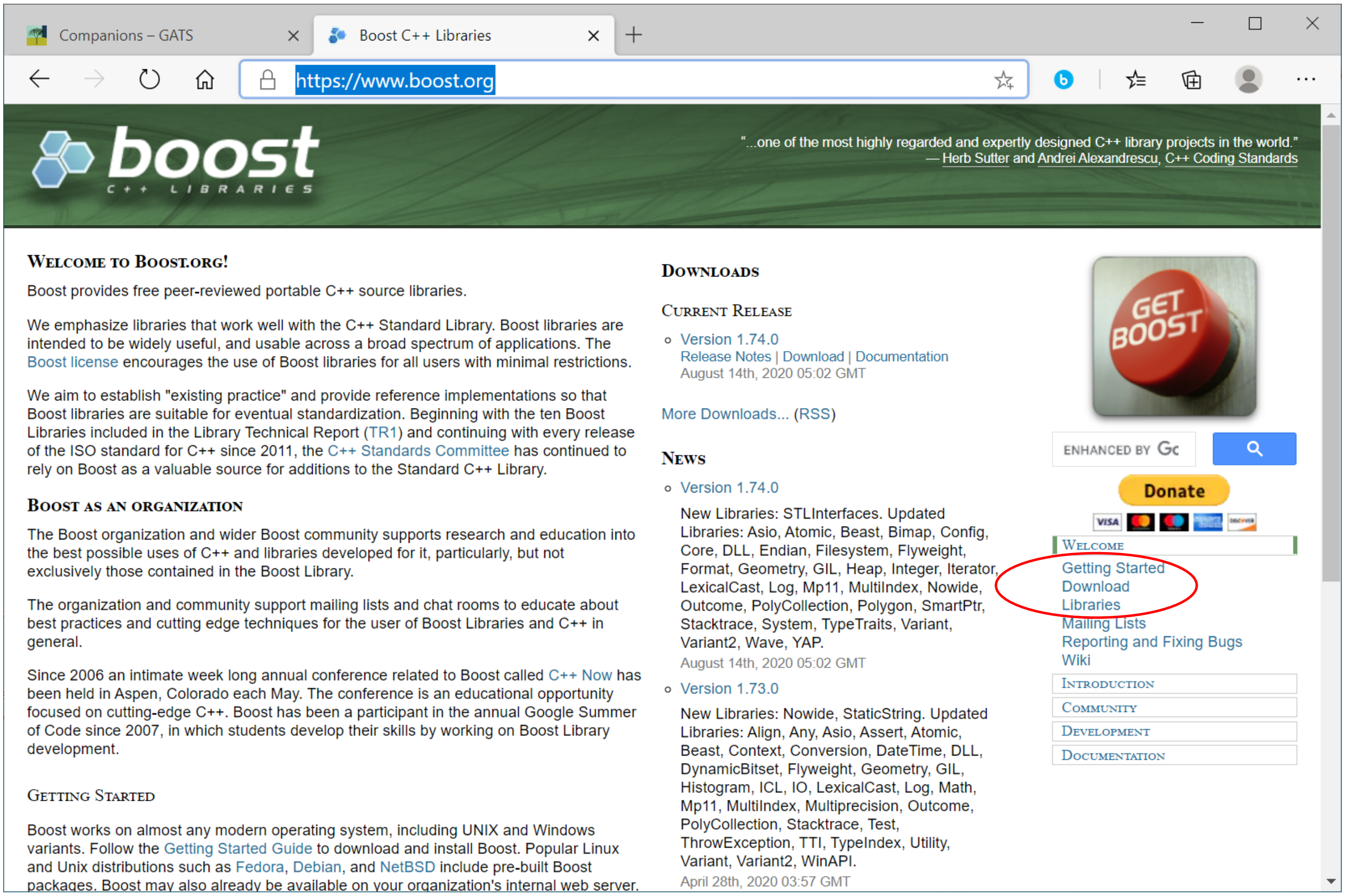
Overview

How to install BOOST for use within Visual Studio 2022.

BOOST updates frequently but the installation process is essentially the same. As such, the version numbers in the images will be 1.74.0 whereas in the text it will 1.83.0. The website may be even newer!

Acquiring the files

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



"...one of the most highly regarded and expertly designed C++ library projects in the world."
— Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

WELCOME TO BOOST.ORG!

Boost provides free peer-reviewed portable C++ source libraries.

We emphasize libraries that work well with the C++ Standard Library. Boost libraries are intended to be widely useful, and usable across a broad spectrum of applications. The Boost license encourages the use of Boost libraries for all users with minimal restrictions.

We aim to establish "existing practice" and provide reference implementations so that Boost libraries are suitable for eventual standardization. Beginning with the ten Boost Libraries included in the Library Technical Report (TR1) and continuing with every release of the ISO standard for C++ since 2011, the C++ Standards Committee has continued to rely on Boost as a valuable source for additions to the Standard C++ Library.

BOOST AS AN ORGANIZATION

The Boost organization and wider Boost community supports research and education into the best possible uses of C++ and libraries developed for it, particularly, but not exclusively those contained in the Boost Library.

The organization and community support mailing lists and chat rooms to educate about best practices and cutting edge techniques for the user of Boost Libraries and C++ in general.

Since 2006 an intimate week long annual conference related to Boost called C++ Now has been held in Aspen, Colorado each May. The conference is an educational opportunity focused on cutting-edge C++. Boost has been a participant in the annual Google Summer of Code since 2007, in which students develop their skills by working on Boost Library development.

GETTING STARTED

Boost works on almost any modern operating system, including UNIX and Windows variants. Follow the Getting Started Guide to download and install Boost. Popular Linux and Unix distributions such as Fedora, Debian, and NetBSD include pre-built Boost packages. Boost may also already be available on your organization's internal web server.

DOWNLOADS

CURRENT RELEASE

- Version 1.74.0
  Release Notes | Download | Documentation
  August 14th, 2020 05:02 GMT

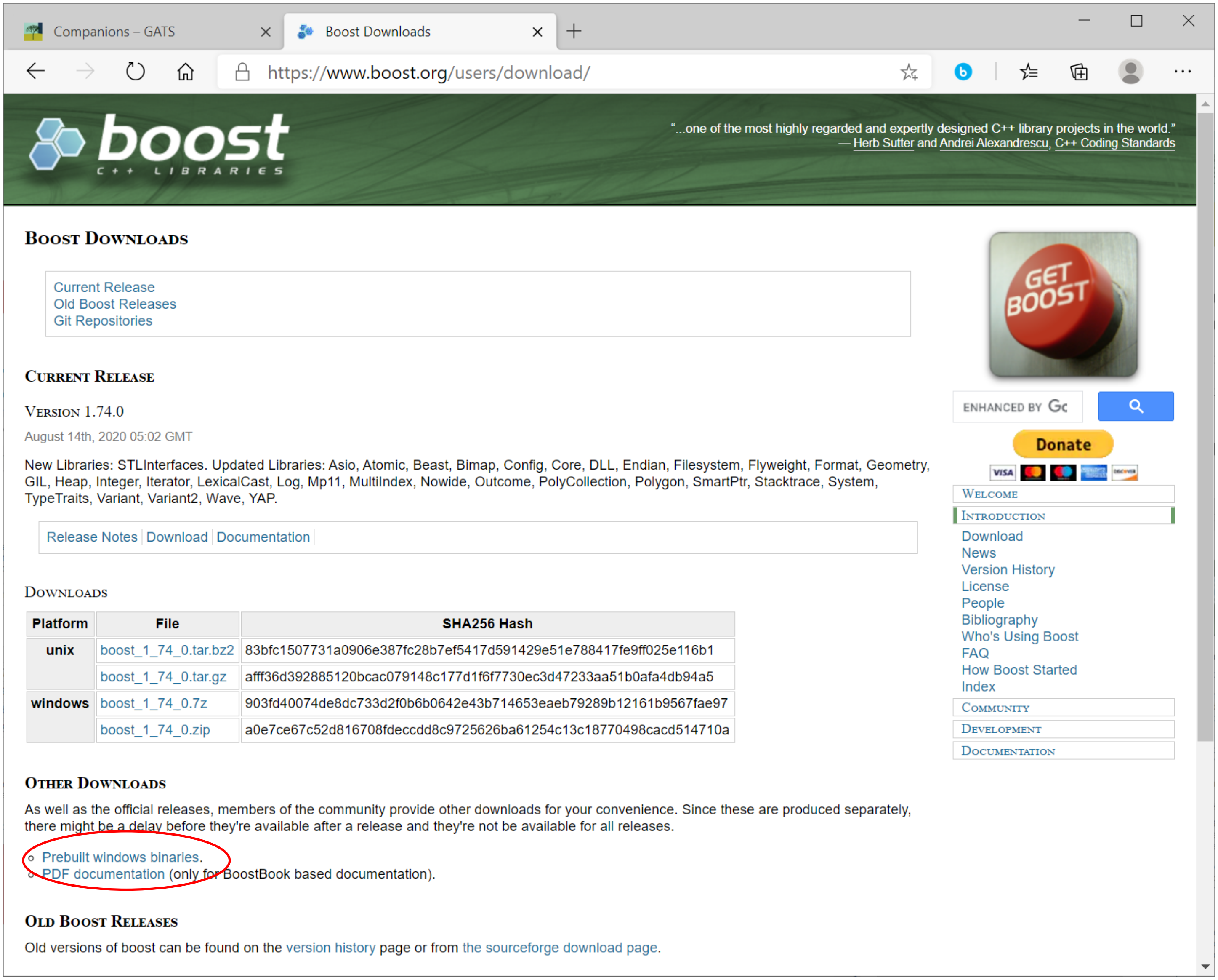
More Downloads... (RSS)

NEWS

- Version 1.74.0
  New Libraries: STLInterfaces. Updated Libraries: Asio, Atomic, Beast, Bimap, Config, Core, DLL, Endian, Filesystem, Flyweight, Format, Geometry, GIL, Heap, Integer, Iterator, LexicalCast, Log, Mp11, MultiIndex, Nowide, Outcome, PolyCollection, Polygon, SmartPtr, Stacktrace, System, TypeTraits, Variant, Variant2, Wave, YAP.
  August 14th, 2020 05:02 GMT
- Version 1.73.0
  New Libraries: Nowide, StaticString. Updated Libraries: Align, Any, Asio, Assert, Atomic, Beast, Context, Conversion, DateTime, DLL, DynamicBitset, Flyweight, Geometry, GIL, Histogram, ICL, IO, LexicalCast, Log, Math, Mp11, MultiIndex, Multiprecision, Outcome, PolyCollection, Stacktrace, Test, ThrowException, TTI, TypeIndex, Utility, Variant, Variant2, WinAPI.
  April 28th, 2020 03:57 GMT

Pick the *prebuilt windows binaries* link.

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



"...one of the most highly regarded and expertly designed C++ library projects in the world."
— Herb Sutter and Andrei Alexandrescu, C++ Coding Standards

Boost Downloads

Current Release
Old Boost Releases
Git Repositories

Current Release

Version 1.74.0

August 14th, 2020 05:02 GMT

New Libraries: STLInterfaces. Updated Libraries: Asio, Atomic, Beast, Bimap, Config, Core, DLL, Endian, Filesystem, Flyweight, Format, Geometry, GIL, Heap, Integer, Iterator, LexicalCast, Log, Mp11, MultiIndex, Nowide, Outcome, PolyCollection, Polygon, SmartPtr, Stacktrace, System, TypeTraits, Variant, Variant2, Wave, YAP.

Release Notes | Download | Documentation |

Downloads

Platform	File	SHA256 Hash
unix	boost_1_74_0.tar.bz2	83bfc1507731a0906e387fc28b7ef5417d591429e51e788417fe9ff025e116b1
	boost_1_74_0.tar.gz	afff36d392885120bcac079148c177d1f6f7730ec3d47233aa51b0afa4db94a5
windows	boost_1_74_0.7z	903fd40074de8dc733d2f0b6b0642e43b714653eaeb79289b12161b9567fae97
	boost_1_74_0.zip	a0e7ce67c52d816708fdeccdd8c9725626ba61254c13c18770498cacd514710a

Other Downloads

As well as the official releases, members of the community provide other downloads for your convenience. Since these are produced separately, there might be a delay before they're available after a release and they're not be available for all releases.

- Prebuilt windows binaries.
- PDF documentation (only for BoostBook based documentation).

Old Boost Releases

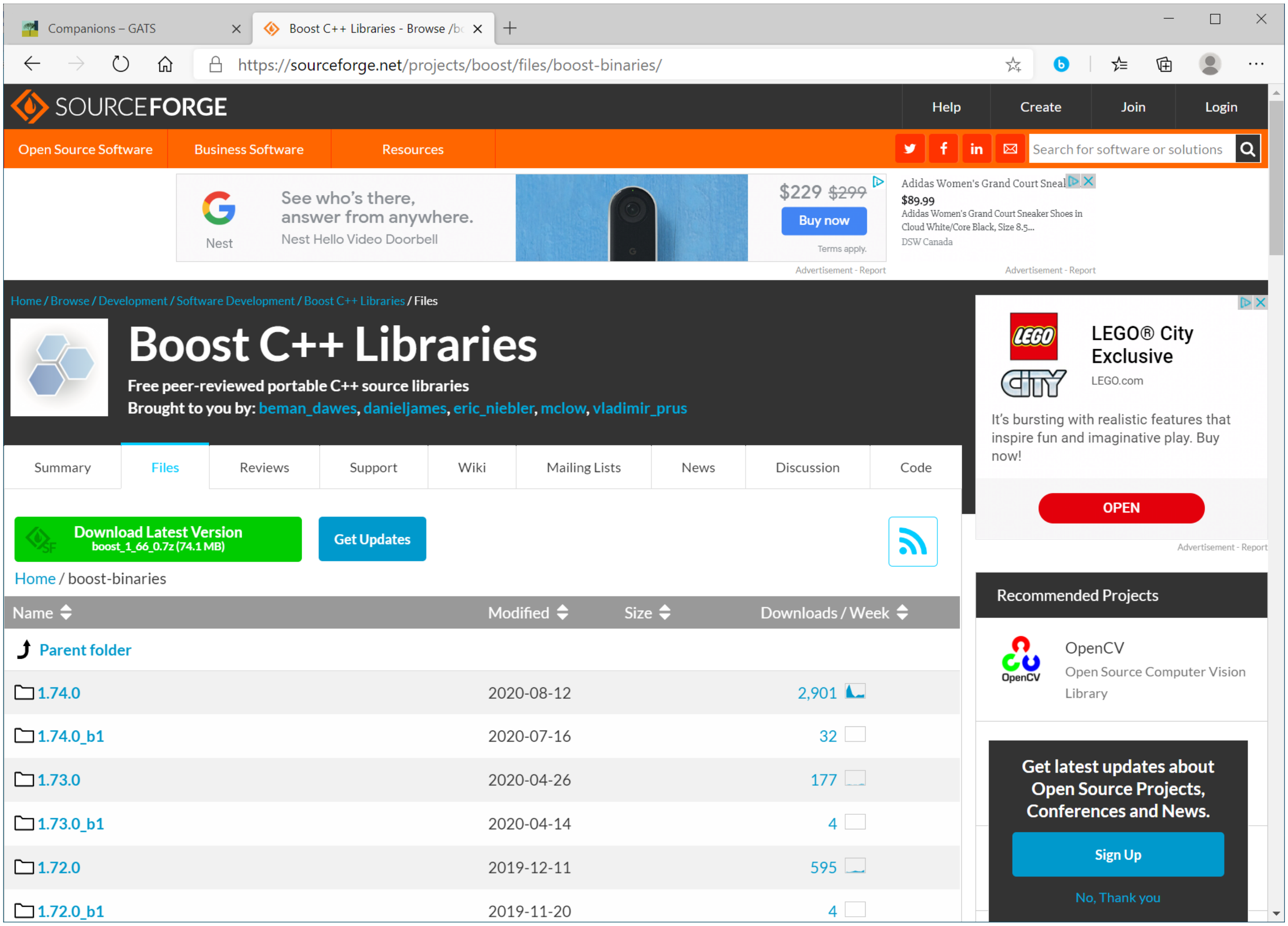
Old versions of boost can be found on the version history page or from the sourceforge download page.

Welcome
Introduction
- Download
- News
- Version History
- License
- People
- Bibliography
- Who's Using Boost
- FAQ
- How Boost Started
- Index

Community
Development
Documentation

You'll now be on SourceForge. Pick the latest *1.83.0* version link.

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


Companions – GATS
Boost C++ Libraries - Browse /bo
https://sourceforge.net/projects/boost/files/boost-binaries/
SOURCEFORGE
Help
Create
Join
Login
Open Source Software
Business Software
Resources
Search for software or solutions

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

Home / boost-binaries

Name	Modified	Size	Downloads / Week
Parent folder			
1.74.0	2020-08-12		2,901
1.74.0_b1	2020-07-16		32
1.73.0	2020-04-26		177
1.73.0_b1	2020-04-14		4
1.72.0	2019-12-11		595
1.72.0_b1	2019-11-20		4

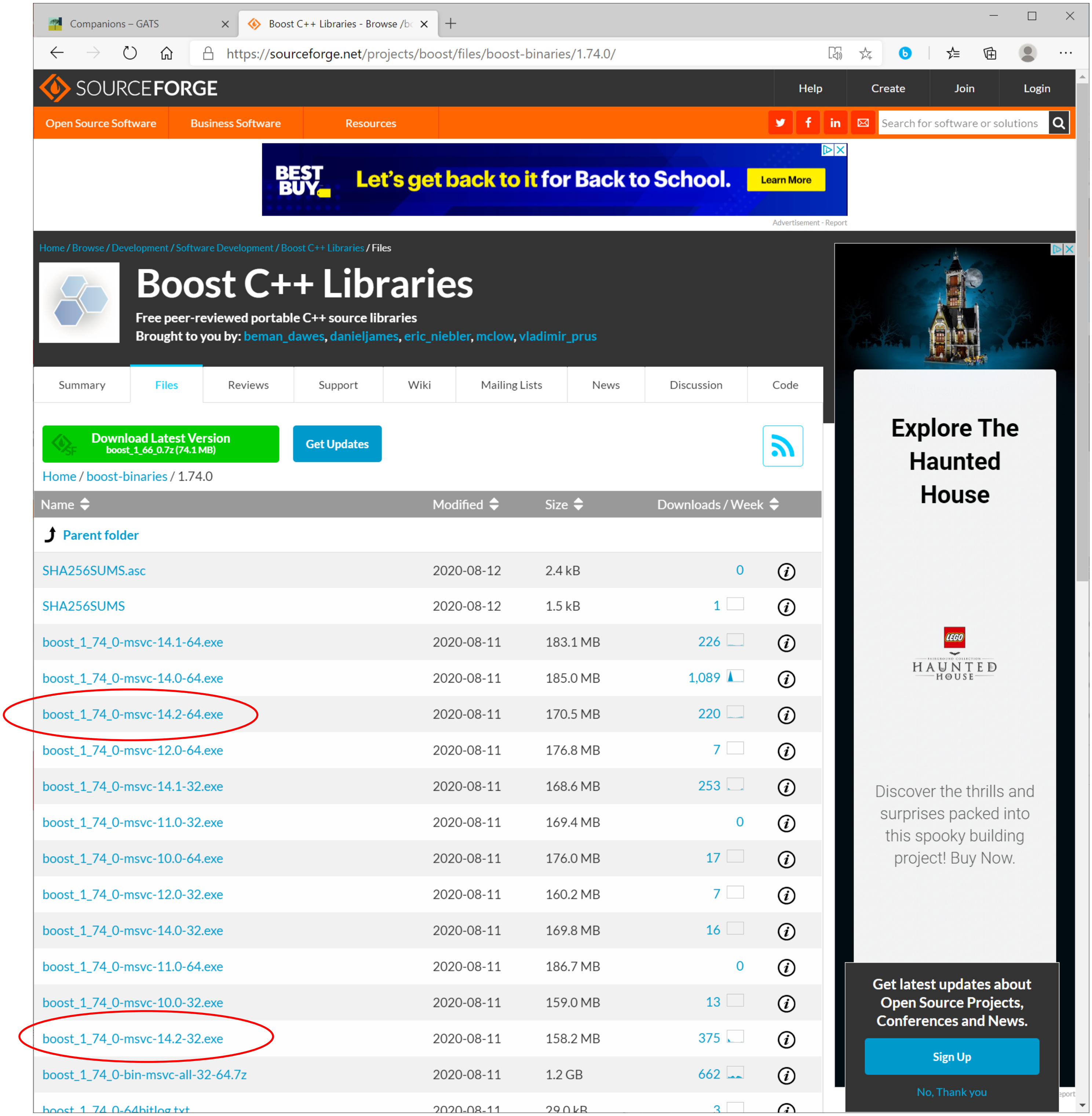
Recommended Projects
OpenCV
Open Source Computer Vision Library
Get latest updates about Open Source Projects, Conferences and News.
Sign Up
No, Thank you

There are two binaries you can install, a 32-bit version, and a 64-bit version. These days you'll likely only need the 64-bit version (Visual C++ 2022 defaults to 64-bit builds). You'll also have to select the release for the current version of Visual Studio 2022. As of this writing, the updated version of Visual Studio 2022 is compiler is **17.7.6**, but the binary format is **14.3**, so that's the version we will use.

https://sourceforge.net/projects/boost/files/boost-binaries/1.83.0/

Download both:

- boost_1_83_0-msvc_14.3-32.exe
- boost_1_83_0-msvc_14.3-64.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

Home / Browse / Development / Software Development / Boost C++ Libraries / Files

Boost C++ Libraries

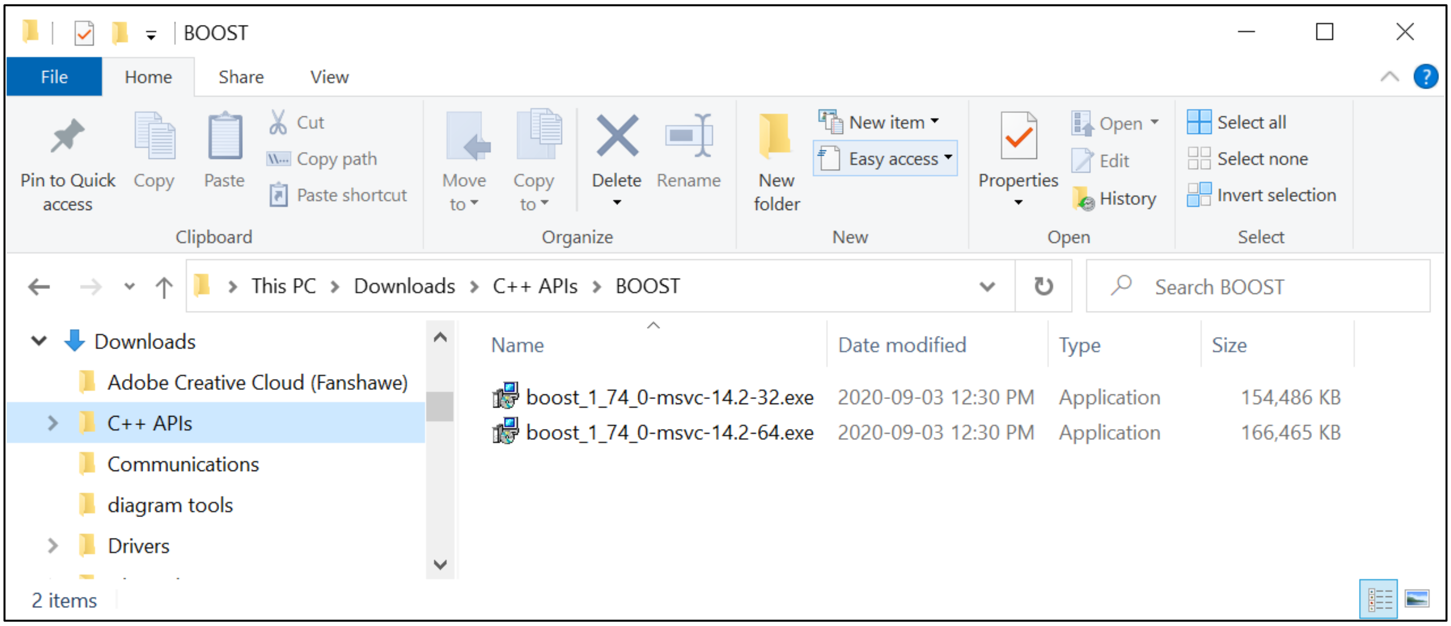
Free peer-reviewed portable C++ source libraries
Brought to you by: beman_dawes, danieljames, eric_niebler, mclow, vladimir_prus

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
boost_1_74_0-64bitlog.txt	2020-08-11	29.0 kB	3

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…



Name	Date modified	Type	Size
boost_1_74_0-msvc-14.2-32.exe	2020-09-03 12:30 PM	Application	154,486 KB
boost_1_74_0-msvc-14.2-64.exe	2020-09-03 12:30 PM	Application	166,465 KB

Install the libraries

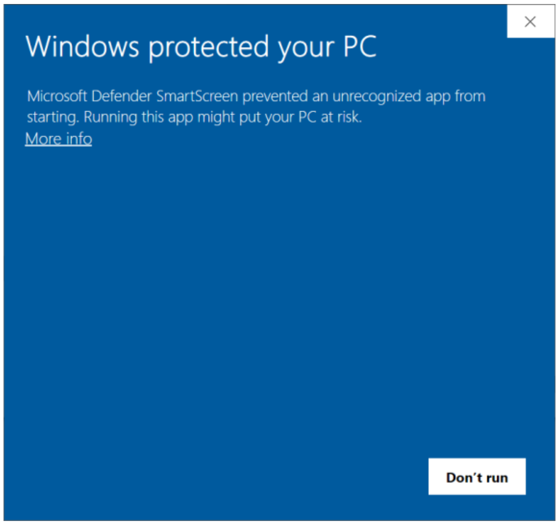
The two files you downloaded are self-extracting archives. You'll need about 4GiB of free memory (ideally on drive C). There are two reasons drive C is the best choice:

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 2 TiB SSD for my primary drive (drive C:) and a slightly slower 4 TiB SDD for my secondary drive (drive D:). Since boost has a huge number of small header files (15,673 in version 1.83.0) 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.

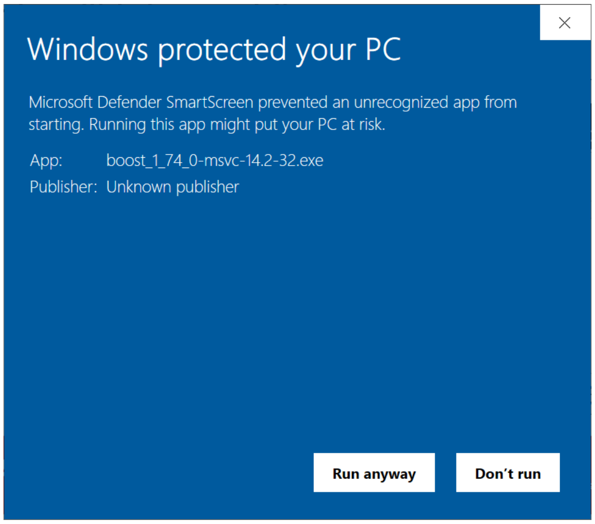
You'll probably see this as soon as you run the installer.



Windows protected your PC

Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

Click More info, you should see:



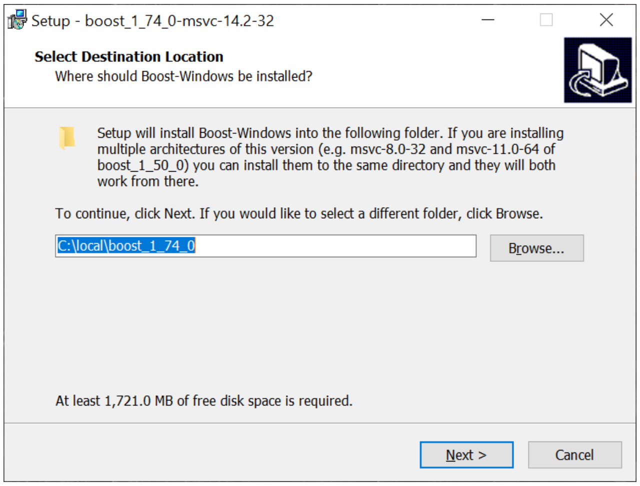
Windows protected your PC

Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

App: boost_1_74_0-msvc-14.2-32.exe
Publisher: Unknown publisher

Click **Run anyway**.

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



Setup - boost_1_74_0-msvc-14.2-32

Select Destination Location

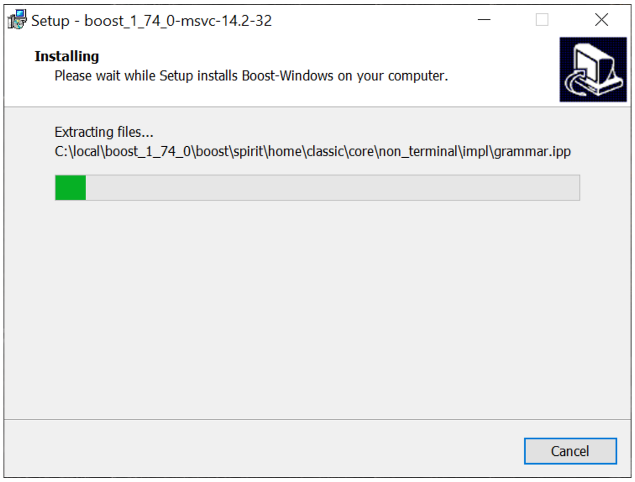
Where should Boost-Windows be installed?

Setup will install Boost-Windows into the following folder. If you are installing multiple architectures of this version (e.g. msvc-8.0-32 and msvc-11.0-64 of boost_1_50_0) you can install them to the same directory and they will both work from there.

To continue, click Next. If you would like to select a different folder, click Browse.

C:\local\boost_1_74_0

At least 1,721.0 MB of free disk space is required.



Setup - boost_1_74_0-msvc-14.2-32

Installing

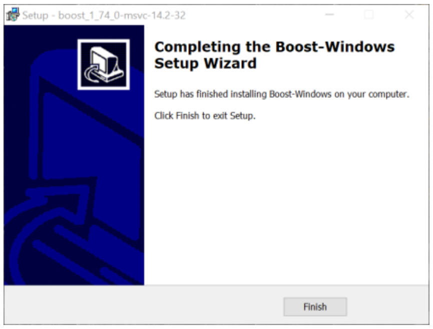
Please wait while Setup installs Boost-Windows on your computer.

Extracting files...

C:\local\boost_1_74_0\boost\spirit\home\classic\core\non_terminal\impl\grammar.ipp

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.

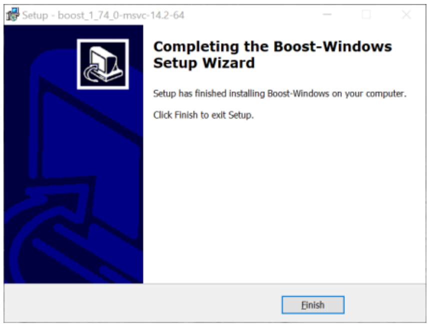


Setup - boost_1_74_0-msvc-14.2-32

Completing the Boost-Windows Setup Wizard

Setup has finished installing Boost-Windows on your computer.

Click Finish to exit Setup.



Setup - boost_1_74_0-msvc-14.2-64

Completing the Boost-Windows Setup Wizard

Setup has finished installing Boost-Windows on your computer.

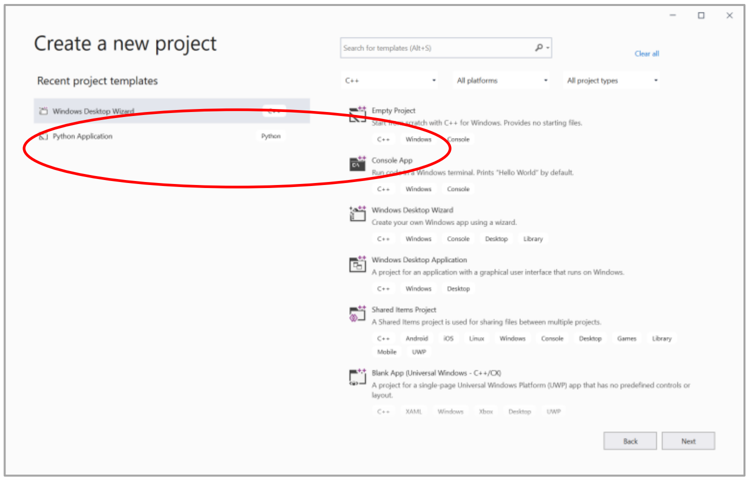
Click Finish to exit Setup.

Setting the Project Properties for Visual C++ with BOOST

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 2022. We'll create a *Hello, BOOST!* application to test our configuration.

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



Create a new project

Recent project templates

Windows Desktop Wizard — C++

Python Application — Python

Search for templates (Alt+S)

C++ · All platforms · All project types

Empty Project
Start from scratch with C++ for Windows. Provides no starting files.
C++ · Windows · Console

Console App
Run code in a Windows terminal. Prints "Hello World" by default.
C++ · Windows · Console

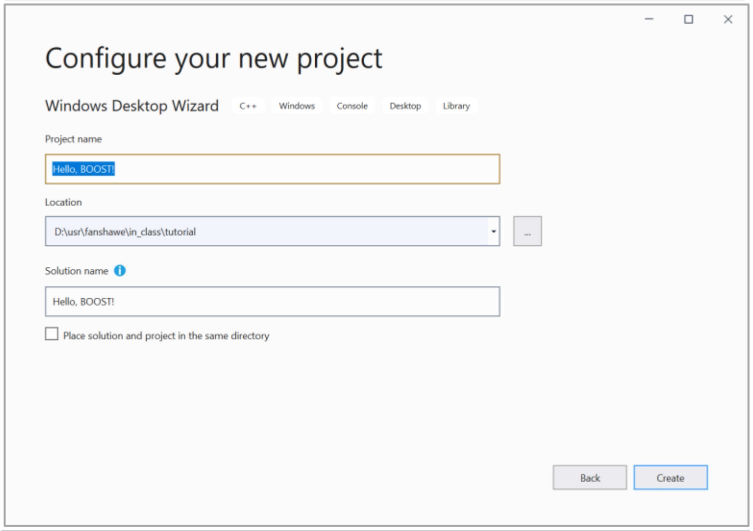
Windows Desktop Wizard
Create your own Windows app using a wizard.
C++ · Windows · Console · Desktop · Library

Windows Desktop Application
A project for an application with a graphical user interface that runs on Windows.
C++ · Windows · Desktop

Shared Items Project
A Shared Items project is used for sharing files between multiple projects.
C++ · Android · iOS · Linux · Windows · Console · Desktop · Games · Library · Mobile · UWP

Blank App (Universal Windows - C++/CX)
A project for a single-page Universal Windows Platform (UWP) app that has no predefined controls or layout.
C++ · XAML · Windows · Xbox · Desktop · UWP

Name the project…



Configure your new project

Windows Desktop Wizard — C++ · Windows · Console · Desktop · Library

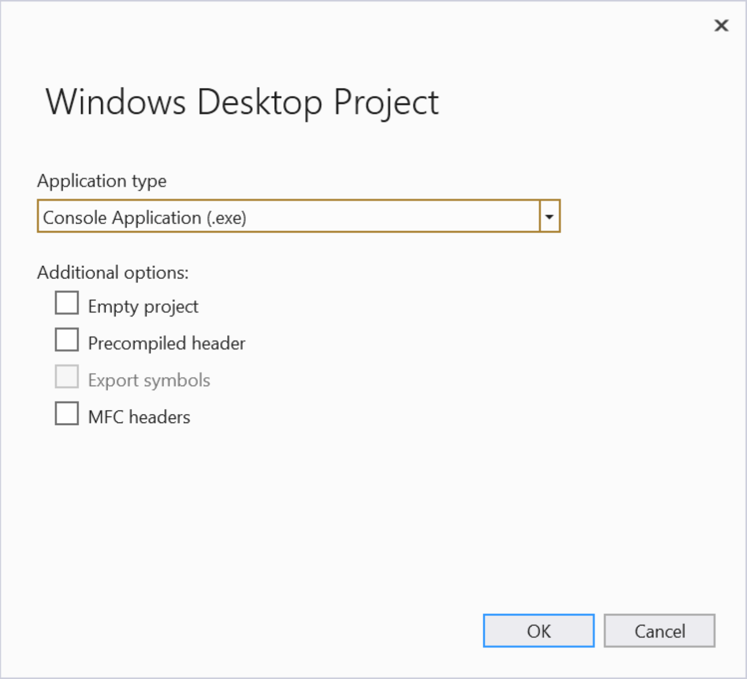
Project name: Hello, BOOST!

Location: D:\usr\fanshawe\in_class\tutorial

Solution name: Hello, BOOST!

☐ Place solution and project in the same directory

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



Windows Desktop Project

Application type: Console Application (.exe)

Additional options:

☐ Empty project

☐ Precompiled header

☐ Export symbols

☐ MFC headers

Ignoring the comments your source code should appear as:

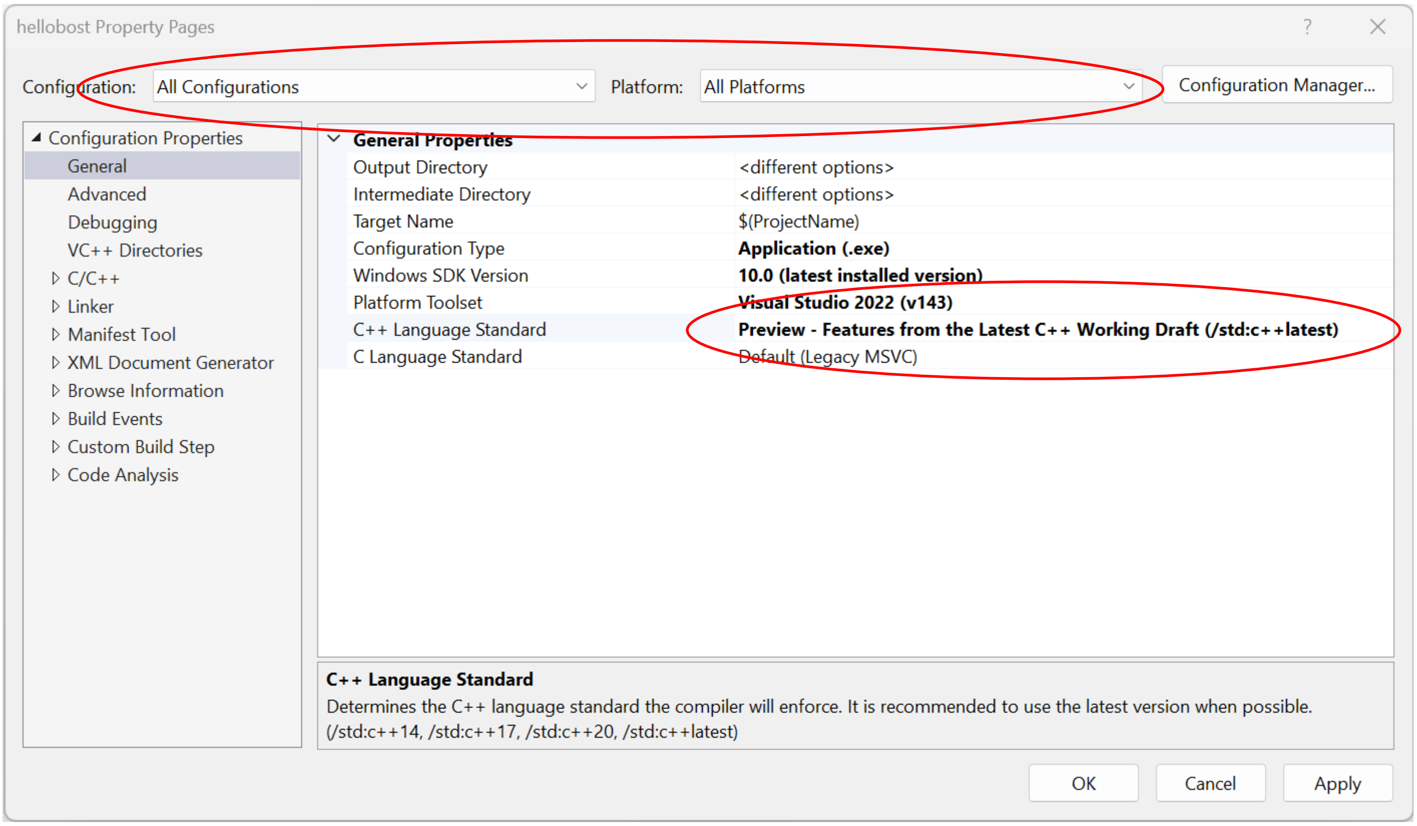
```
#include <iostream>

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("1234567890123456789012345678901234567890");
}
```

Configuring Boost for one project only: (easiest)

Open the project properties for your solution (the wrench icon, or menu Project → Properties).
Let's configure for the latest version of C++. Make certain that you have 'All Configurations' and 'All Platforms' selected.



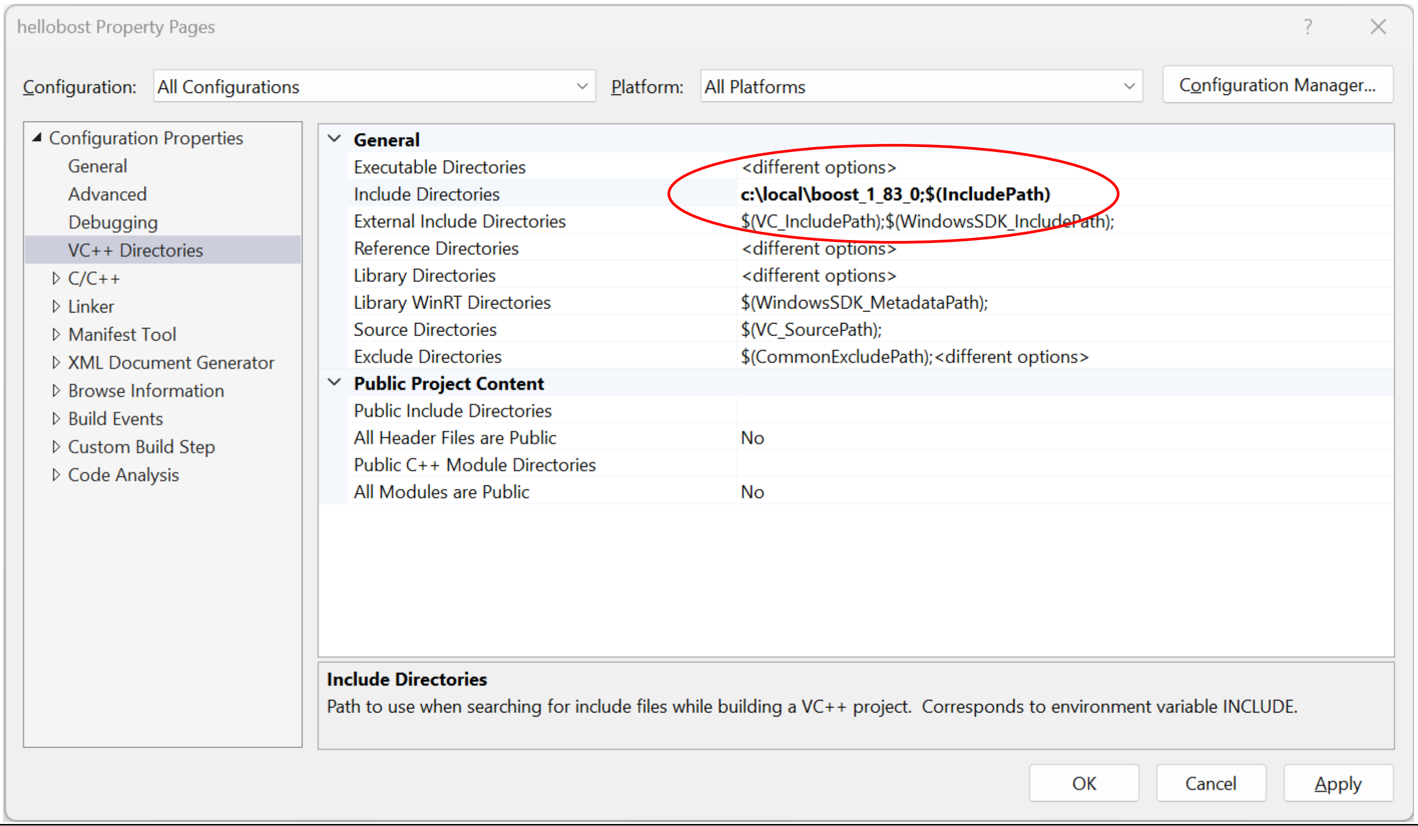
Configuration: All Configurations
Platform: All Platforms

General Properties

Output Directory	<different options>
Intermediate Directory	<different options>
Target Name	$(ProjectName)
Configuration Type	**Application (.exe)**
Windows SDK Version	**10.0 (latest installed version)**
Platform Toolset	**Visual Studio 2022 (v143)**
C++ Language Standard	**Preview - Features from the Latest C++ Working Draft (/std:c++latest)**
C Language Standard	Default (Legacy MSVC)

C++ Language Standard
Determines the C++ language standard the compiler will enforce. It is recommended to use the latest version when possible. (/std:c++14, /std:c++17, /std:c++20, /std:c++latest)

Next set the include directories …



Configuration: All Configurations
Platform: All Platforms

General

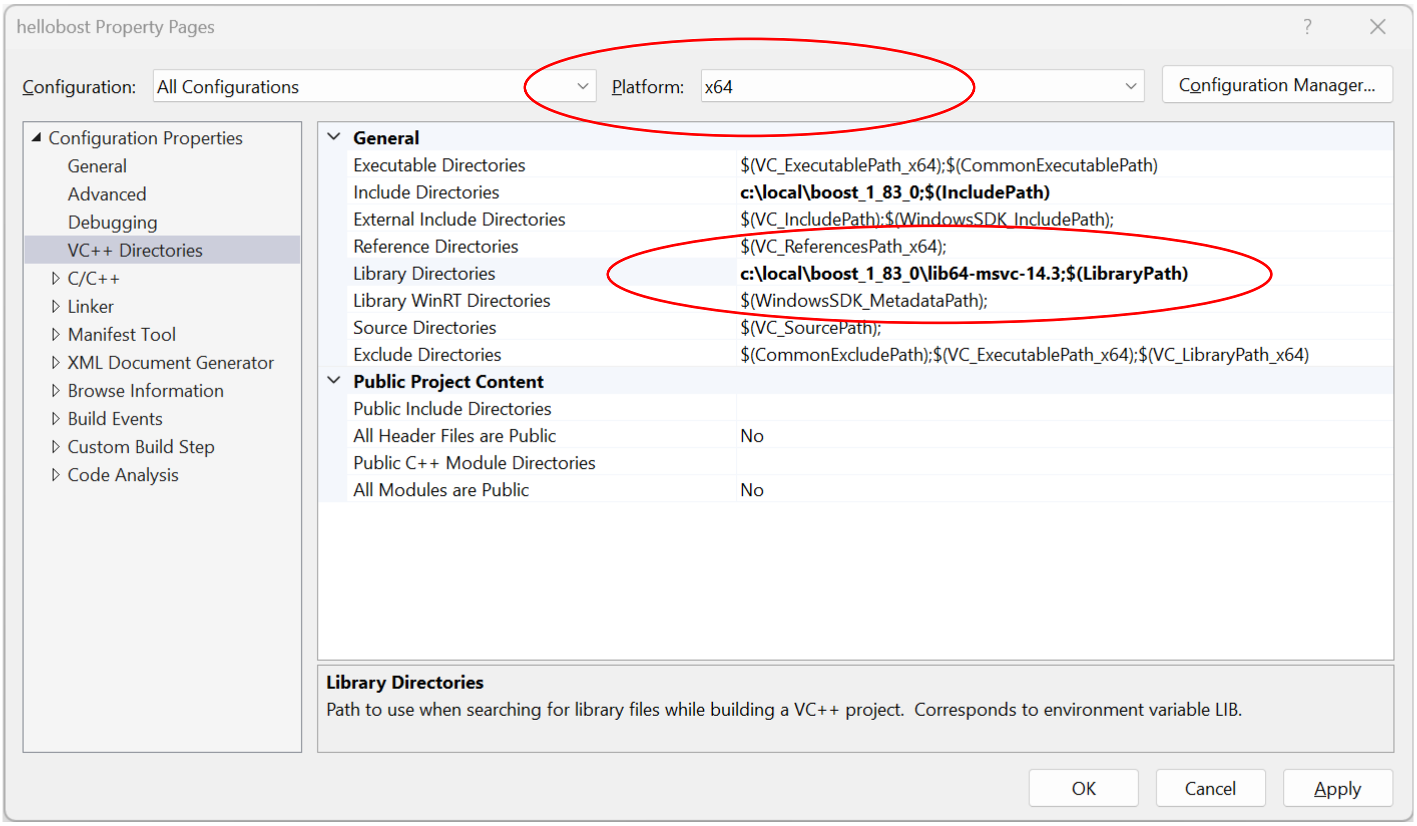
Executable Directories	<different options>
Include Directories	**c:\local\boost_1_83_0;$(IncludePath)**
External Include Directories	$(VC_IncludePath);$(WindowsSDK_IncludePath);
Reference Directories	<different options>
Library Directories	<different options>
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);<different options>

Public Project Content

Public Include Directories	
All Header Files are Public	No
Public C++ Module Directories	
All Modules are Public	No

Include Directories
Path to use when searching for include files while building a VC++ project. Corresponds to environment variable INCLUDE.

Then set the library directories (once for each platform).
64-bit:



Configuration: All Configurations
Platform: x64

General

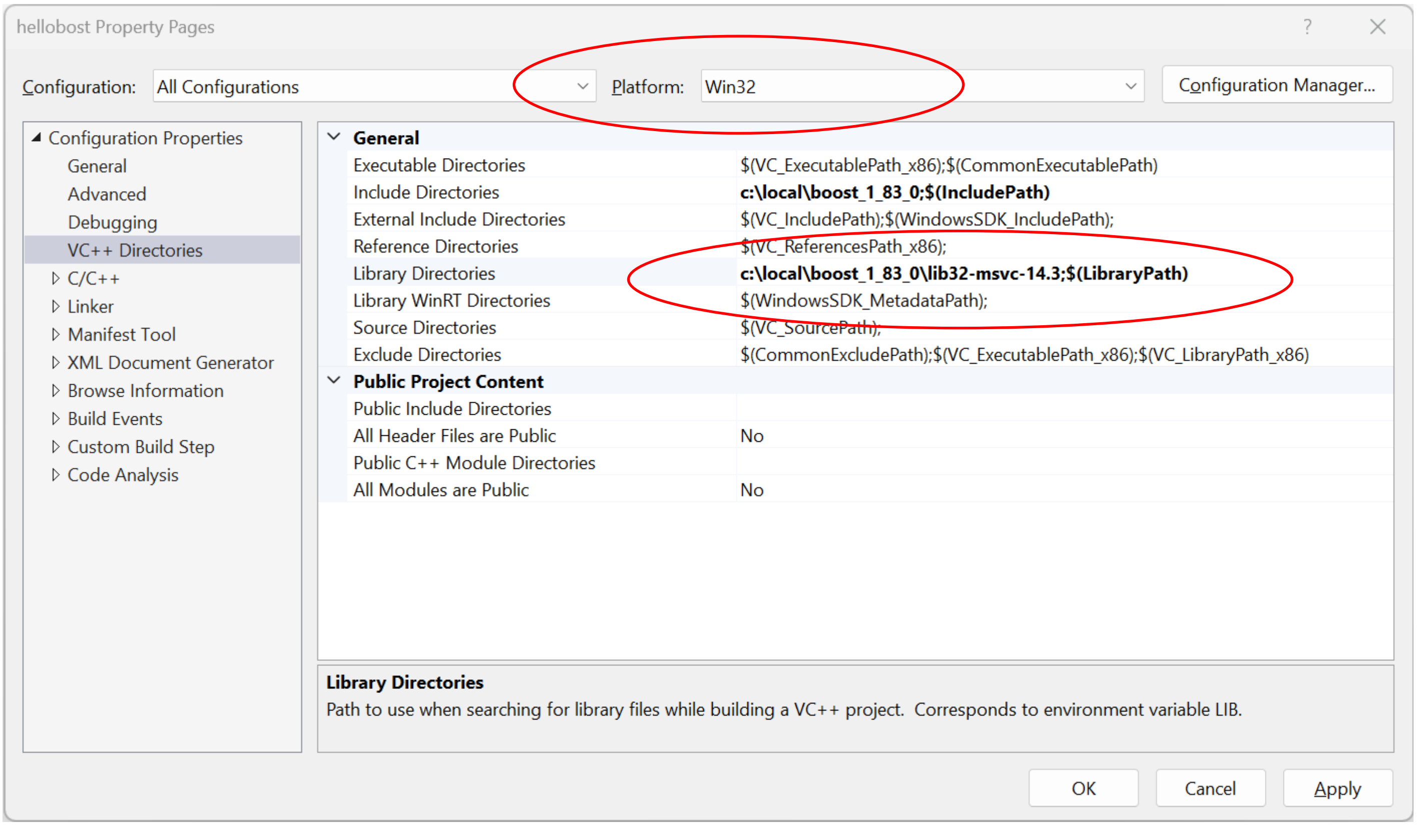
Executable Directories	$(VC_ExecutablePath_x64);$(CommonExecutablePath)
Include Directories	**c:\local\boost_1_83_0;$(IncludePath)**
External Include Directories	$(VC_IncludePath);$(WindowsSDK_IncludePath);
Reference Directories	$(VC_ReferencesPath_x64);
Library Directories	**c:\local\boost_1_83_0\lib64-msvc-14.3;$(LibraryPath)**
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x64);$(VC_LibraryPath_x64)

Public Project Content

Public Include Directories	
All Header Files are Public	No
Public C++ Module Directories	
All Modules are Public	No

Library Directories
Path to use when searching for library files while building a VC++ project. Corresponds to environment variable LIB.

32-bit:



Configuration: All Configurations
Platform: Win32

General

Executable Directories	$(VC_ExecutablePath_x86);$(CommonExecutablePath)
Include Directories	**c:\local\boost_1_83_0;$(IncludePath)**
External Include Directories	$(VC_IncludePath);$(WindowsSDK_IncludePath);
Reference Directories	$(VC_ReferencesPath_x86);
Library Directories	**c:\local\boost_1_83_0\lib32-msvc-14.3;$(LibraryPath)**
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x86);$(VC_LibraryPath_x86)

Public Project Content

Public Include Directories	
All Header Files are Public	No
Public C++ Module Directories	
All Modules are Public	No

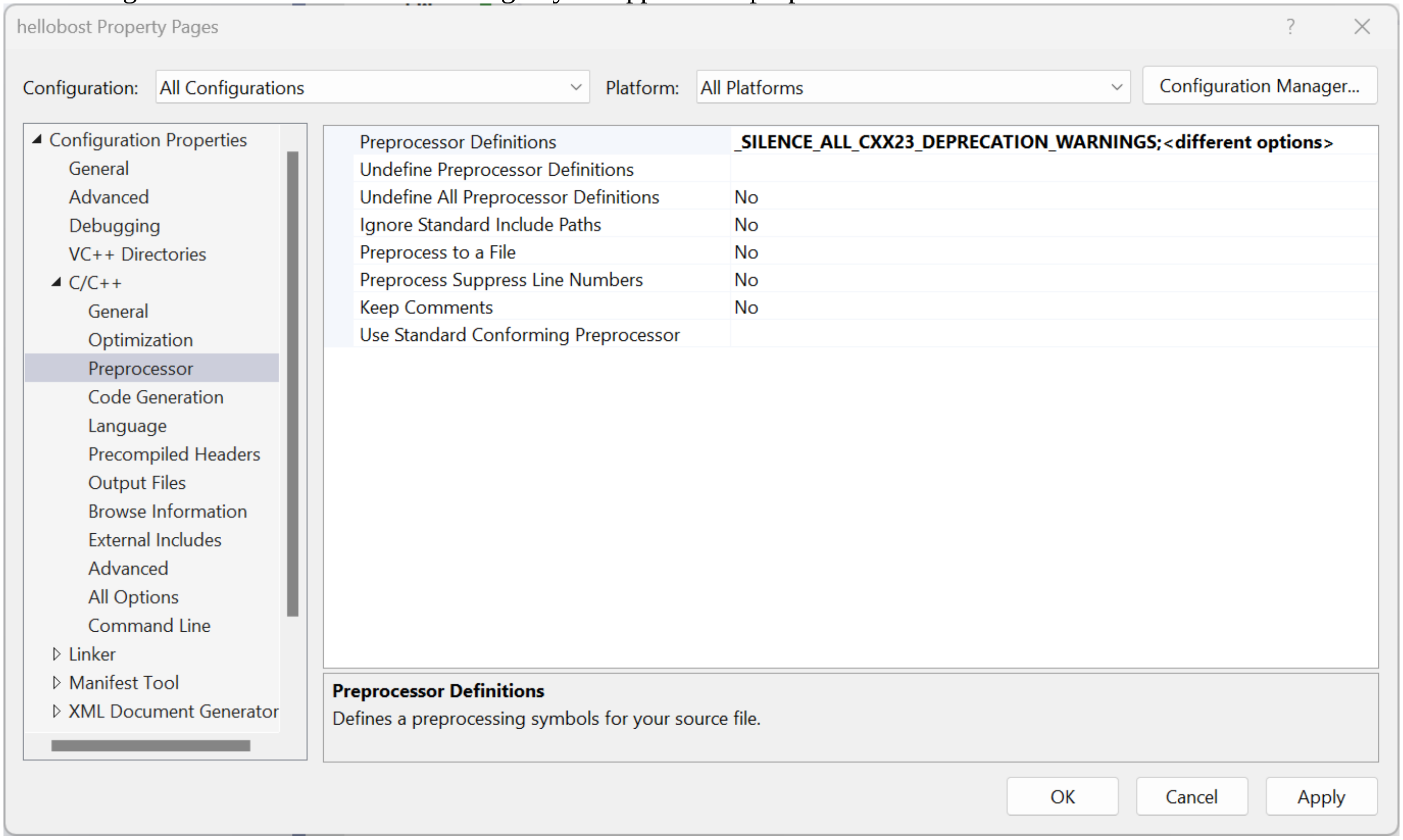
Library Directories
Path to use when searching for library files while building a VC++ project. Corresponds to environment variable LIB.

If you are not using property sheets, you can skip to the next section.

Compiling Boost with C++ 23

Boost 1.83.0 isn't quite ready for C++ 23. It still uses some deprecated features and Visual C++ will complain!

Until Boost catches up with the standard, we can block the compiler from issuing a torrent of errors and warnings about boost using old features. Add the following to your application properties:



hellobost Property Pages

Configuration: All Configurations   Platform: All Platforms

Property	Value
Preprocessor Definitions	_SILENCE_ALL_CXX23_DEPRECATION_WARNINGS;<different options>
Undefine Preprocessor Definitions	
Undefine All Preprocessor Definitions	No
Ignore Standard Include Paths	No
Preprocess to a File	No
Preprocess Suppress Line Numbers	No
Keep Comments	No
Use Standard Conforming Preprocessor	

Preprocessor Definitions
Defines a preprocessing symbols for your source file.

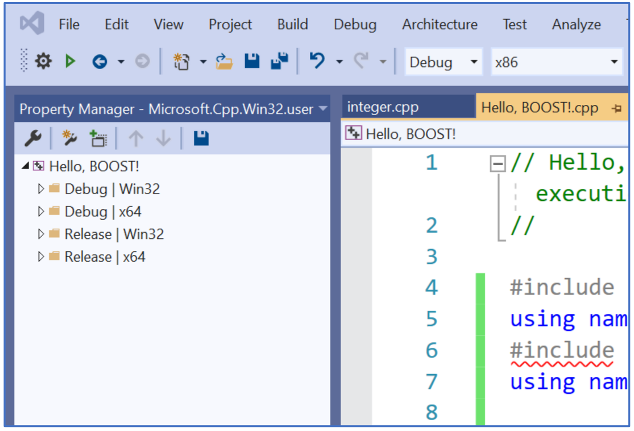
Your BOOST/C++ program should now compile. The rest of the document is for a more advanced, but unnecessary configuration.

Configuring Boost using a Property Page (32-bit)

You can skip this part if you have used the previous method.

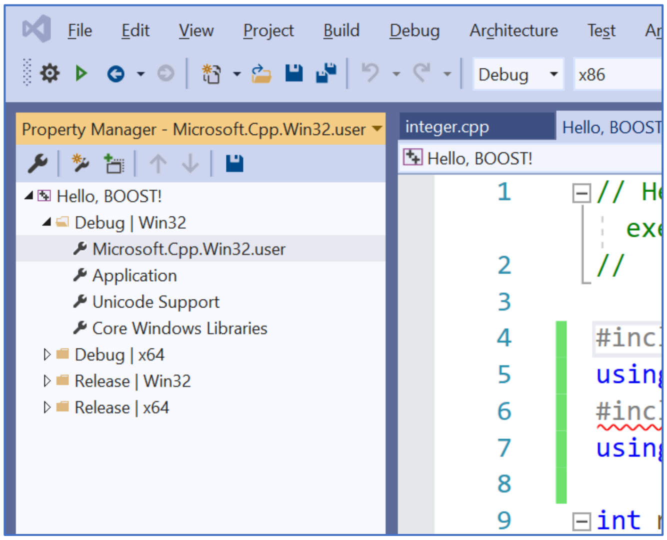
Add a new *reusable* project property sheet (32-bit)

Open the Property Manager   Menu: View → Property Manager



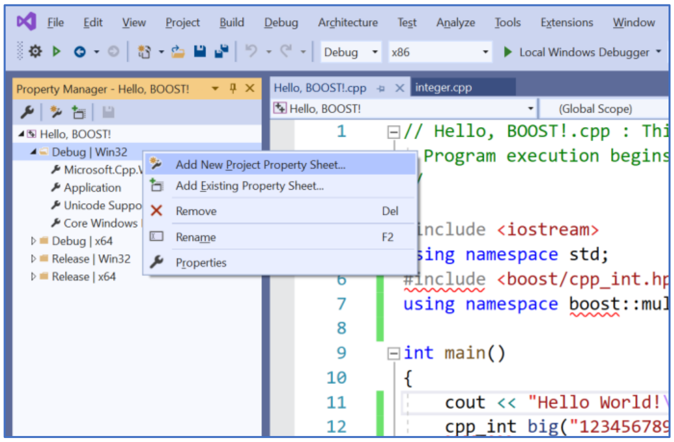
```
// Hello,
   executi
//

#include
using nam
#include
using nam
```

Expand the Debug | Win32 folder



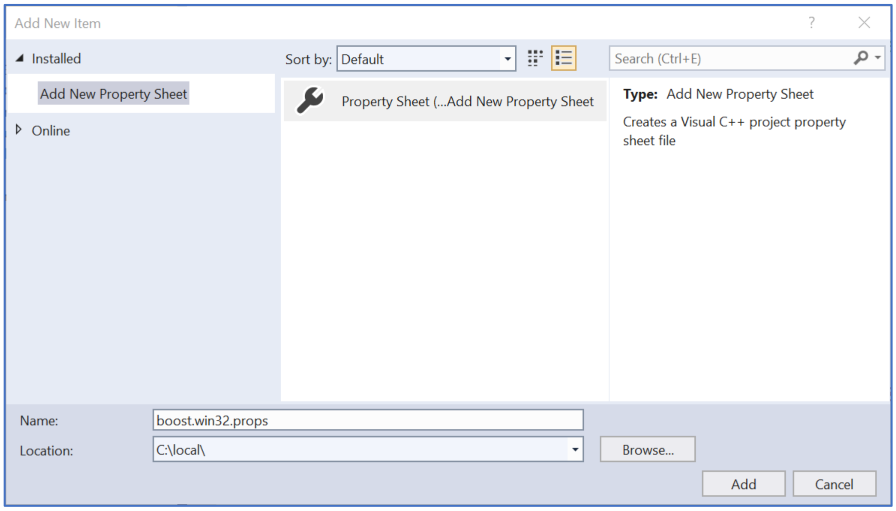
```
// He
  exe
//

#incl
using
#incl
using

int
```

Add a new project property sheet



```
// Hello, BOOST!.cpp : Thi
 Program execution begins

include <iostream>
sing namespace std;
#include <boost/cpp_int.hp
using namespace boost::mul

int main()
{
    cout << "Hello World!\
    cpp_int big("123456789
```

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



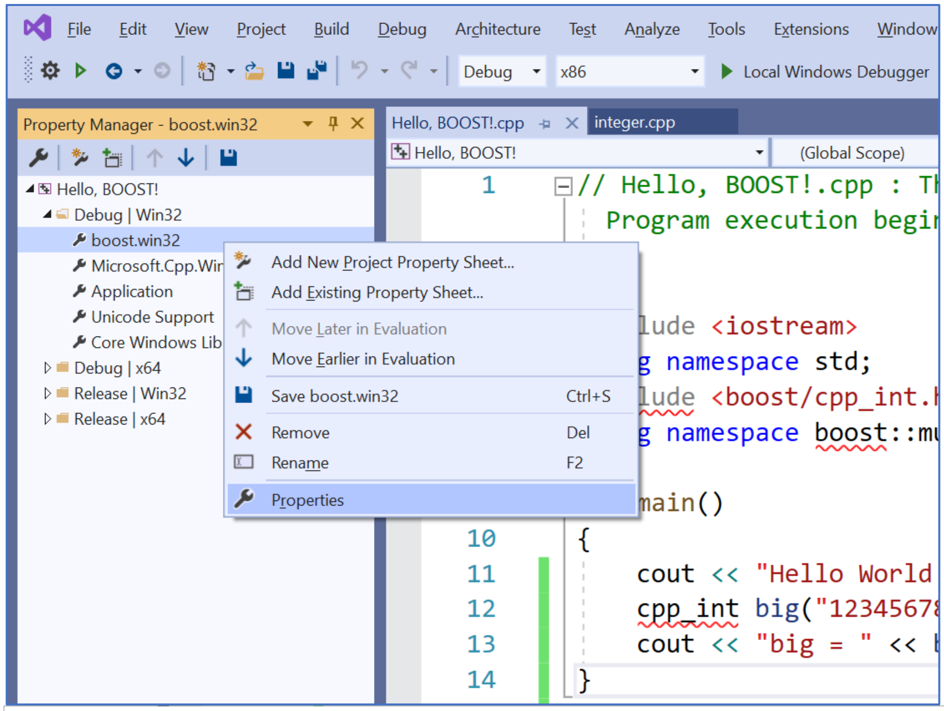
Type: Add New Property Sheet

Creates a Visual C++ project property sheet file

Name: boost.win32.props

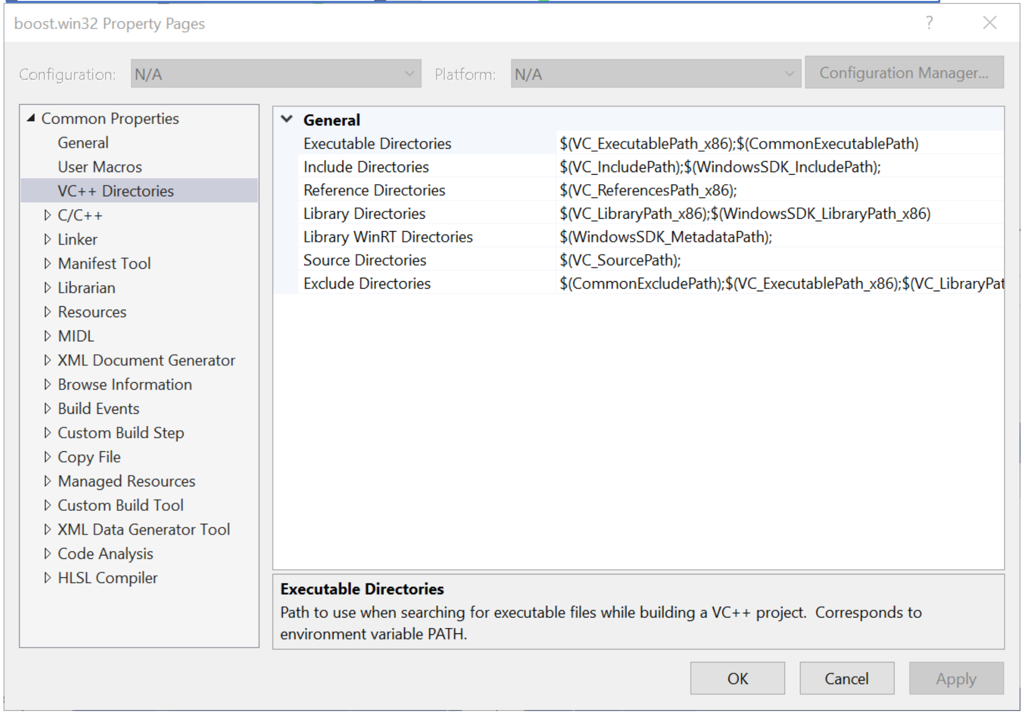
Location: C:\local\

Open the property sheet.



```
// Hello, BOOST!.cpp : Th
  Program execution begi

lude <iostream>
g namespace std;
lude <boost/cpp_int.h
g namespace boost::mu

main()
{
    cout << "Hello World
    cpp_int big("12345678
    cout << "big = " << b
}
```

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



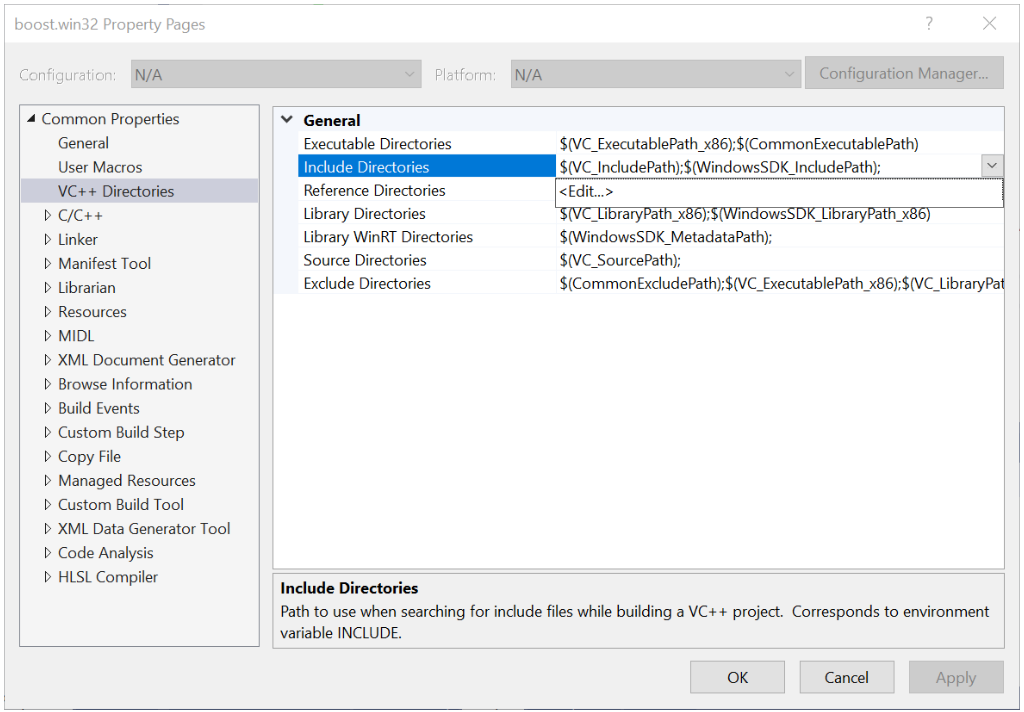
General

Executable Directories	$(VC_ExecutablePath_x86);$(CommonExecutablePath)
Include Directories	$(VC_IncludePath);$(WindowsSDK_IncludePath);
Reference Directories	$(VC_ReferencesPath_x86);
Library Directories	$(VC_LibraryPath_x86);$(WindowsSDK_LibraryPath_x86)
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x86);$(VC_LibraryPat

Executable Directories

Path to use when searching for executable files while building a VC++ project. Corresponds to environment variable PATH.

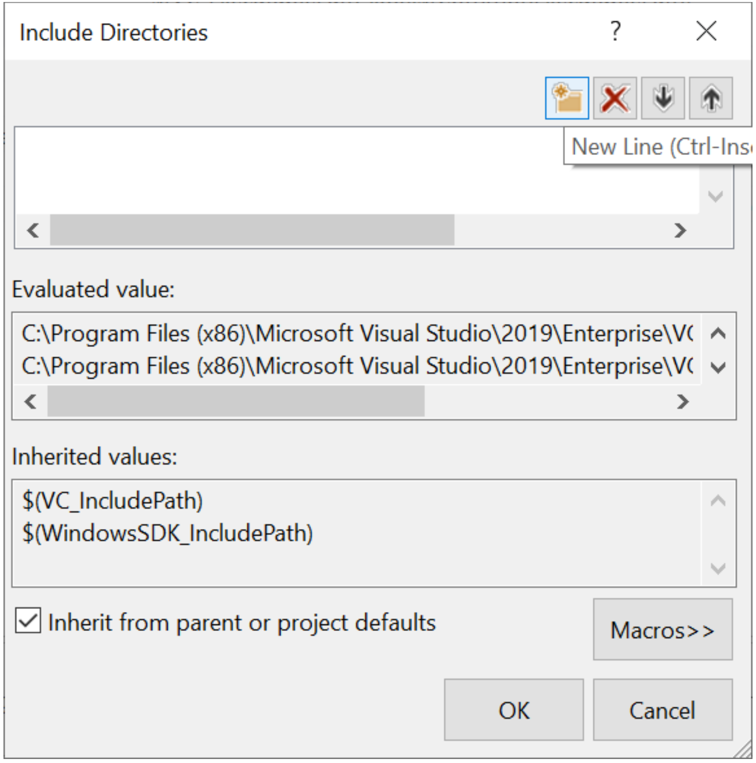
Edit the include paths.



General	
Executable Directories	$(VC_ExecutablePath_x86);$(CommonExecutablePath)
Include Directories	$(VC_IncludePath);$(WindowsSDK_IncludePath);
Reference Directories	<Edit...>
Library Directories	$(VC_LibraryPath_x86);$(WindowsSDK_LibraryPath_x86)
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x86);$(VC_LibraryPat

Include Directories
Path to use when searching for include files while building a VC++ project. Corresponds to environment variable INCLUDE.

Add a new line…



Evaluated value:

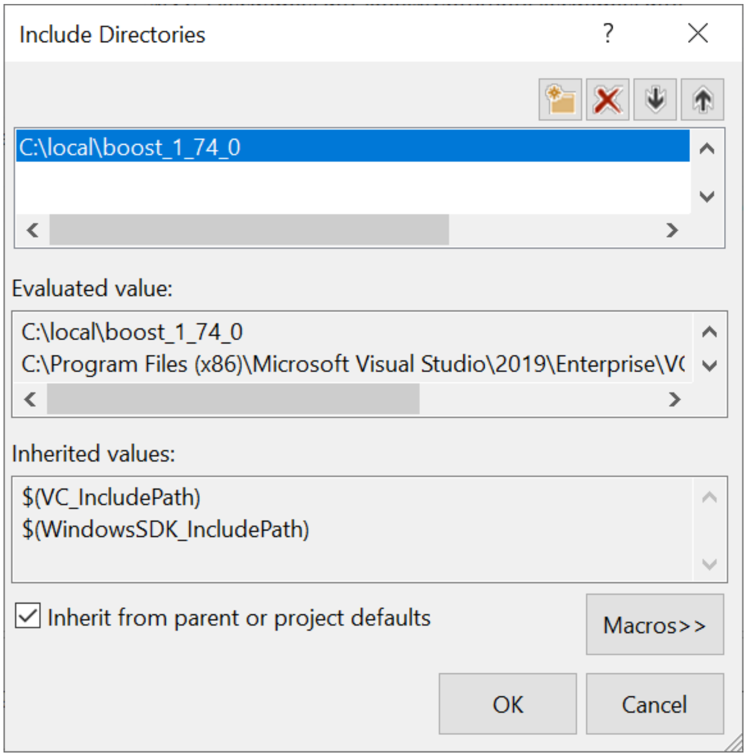
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\V(
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\V(

Inherited values:

$(VC_IncludePath)
$(WindowsSDK_IncludePath)

☑ Inherit from parent or project defaults

Add the boost main folder



C:\local\boost_1_74_0

Evaluated value:

C:\local\boost_1_74_0
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\V(

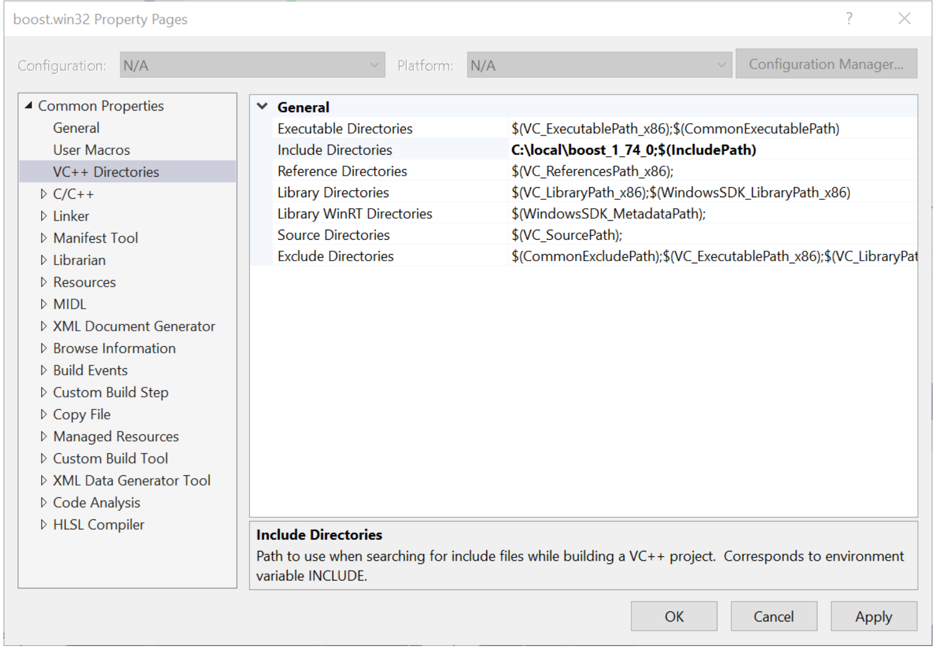
Inherited values:

$(VC_IncludePath)
$(WindowsSDK_IncludePath)

☑ Inherit from parent or project defaults

You should see…

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



Configuration: N/A Platform: N/A

General

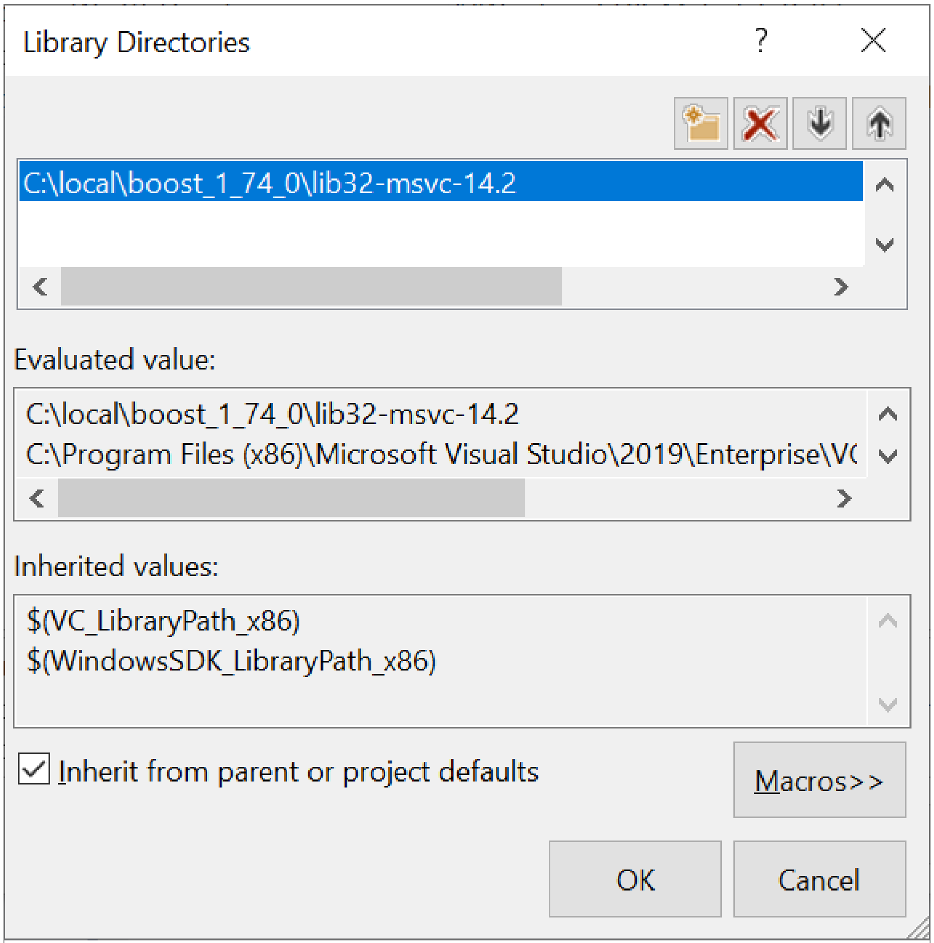
Executable Directories	$(VC_ExecutablePath_x86);$(CommonExecutablePath)
Include Directories	**C:\local\boost_1_74_0;$(IncludePath)**
Reference Directories	$(VC_ReferencesPath_x86);
Library Directories	$(VC_LibraryPath_x86);$(WindowsSDK_LibraryPath_x86)
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x86);$(VC_LibraryPat

Include Directories
Path to use when searching for include files while building a VC++ project. Corresponds to environment variable INCLUDE.

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



Library Directories

C:\local\boost_1_74_0\lib32-msvc-14.2

Evaluated value:

C:\local\boost_1_74_0\lib32-msvc-14.2
C:\Program Files (x86)\Microsoft Visual Studio\2019\Enterprise\VC

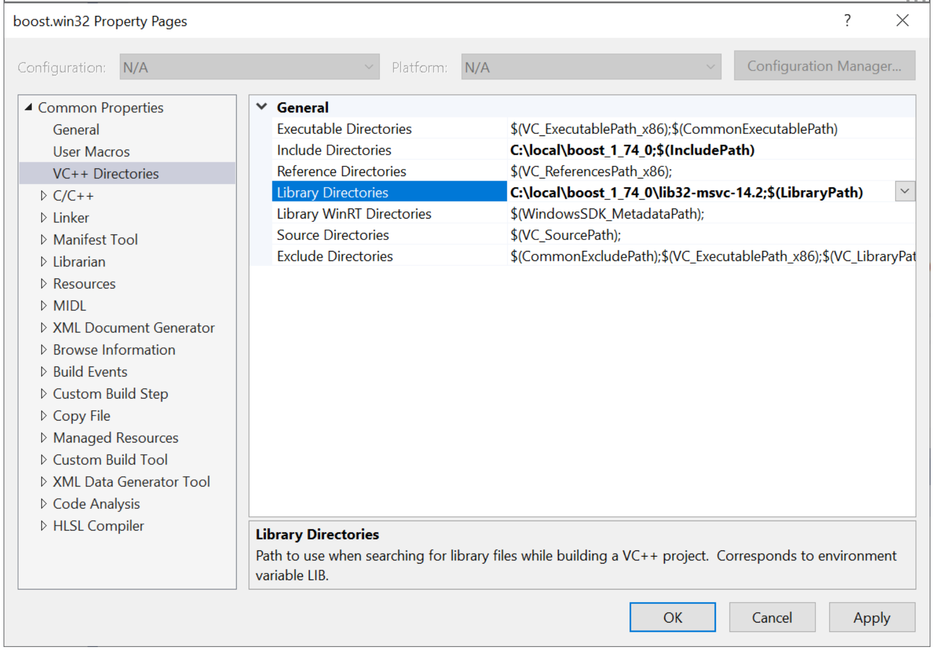
Inherited values:

$(VC_LibraryPath_x86)
$(WindowsSDK_LibraryPath_x86)

☑ Inherit from parent or project defaults

You should see…

Click OK



boost.win32 Property Pages

Configuration: N/A Platform: N/A

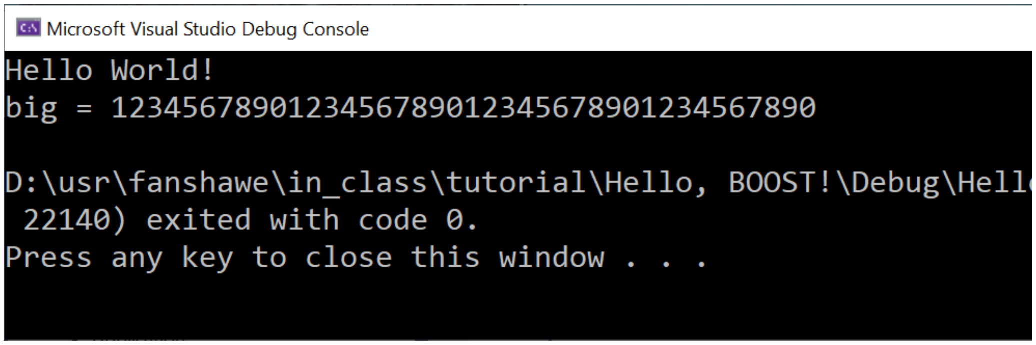
General

Executable Directories	$(VC_ExecutablePath_x86);$(CommonExecutablePath)
Include Directories	**C:\local\boost_1_74_0;$(IncludePath)**
Reference Directories	$(VC_ReferencesPath_x86);
Library Directories	**C:\local\boost_1_74_0\lib32-msvc-14.2;$(LibraryPath)**
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x86);$(VC_LibraryPat

Library Directories
Path to use when searching for library files while building a VC++ project. Corresponds to environment variable LIB.

Test the configuration…

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



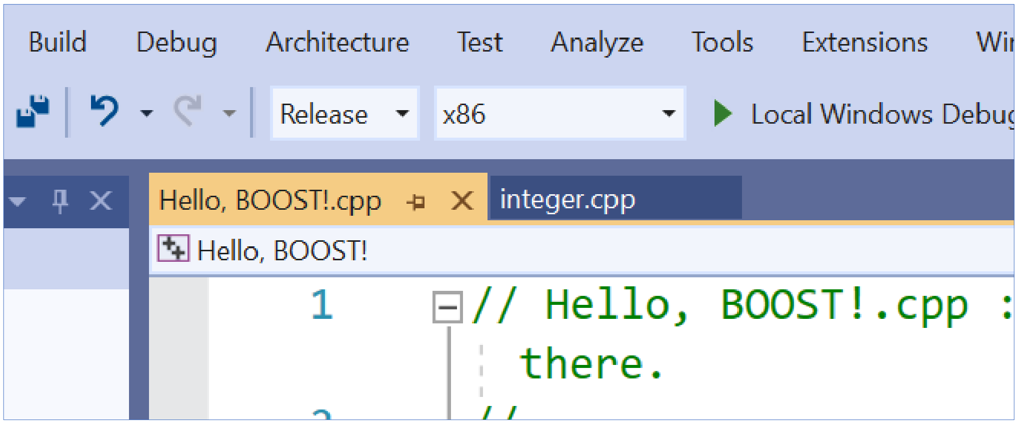
```
Hello World!
big = 1234567890123456789012345678901234567890

D:\usr\fanshawe\in_class\tutorial\Hello, BOOST!\Debug\Hell
 22140) exited with code 0.
Press any key to close this window . . .
```

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.

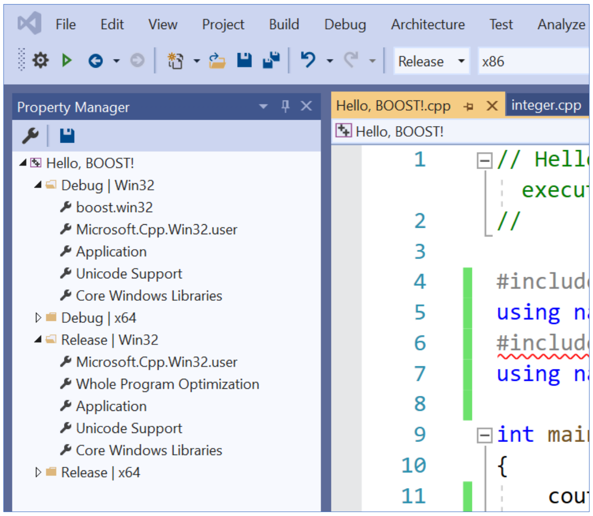
Switch to release mode.

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



```
// Hello, BOOST!.cpp :
  there.
```

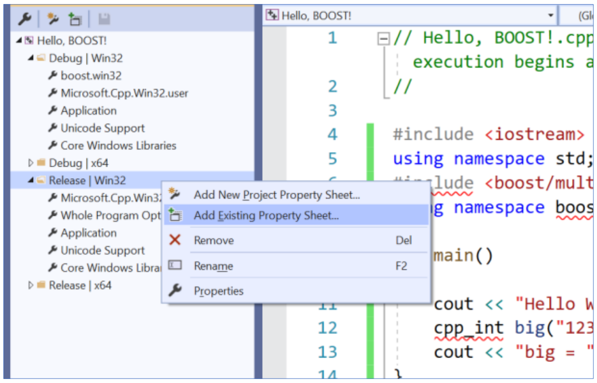
Expand the Release | Win32 tab in the Property Manager



```
Hello, BOOST!
  Debug | Win32
    boost.win32
    Microsoft.Cpp.Win32.user
    Application
    Unicode Support
    Core Windows Libraries
  Debug | x64
  Release | Win32
    Microsoft.Cpp.Win32.user
    Whole Program Optimization
    Application
    Unicode Support
    Core Windows Libraries
  Release | x64
```

```
// Hell
  execu
//

#includ
using n
#includ
using n

int mai
{
    cou
```

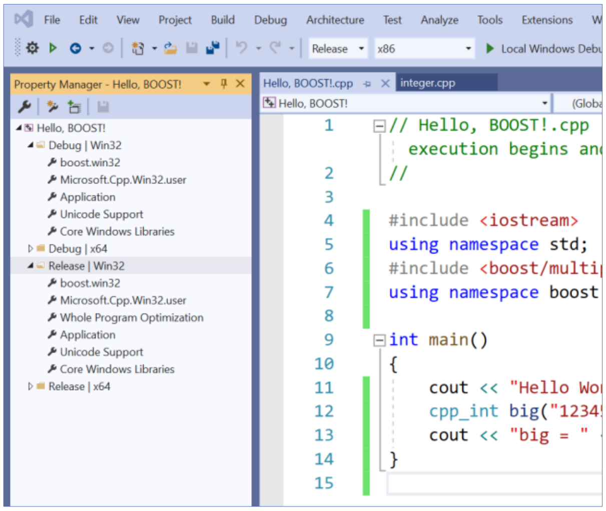
Add the existing property sheet.



```
// Hello, BOOST!.cpp
  execution begins a
//

#include <iostream>
using namespace std;
#include <boost/mult
g namespace boos

main()

  cout << "Hello W
  cpp_int big("123
  cout << "big = "
}
```

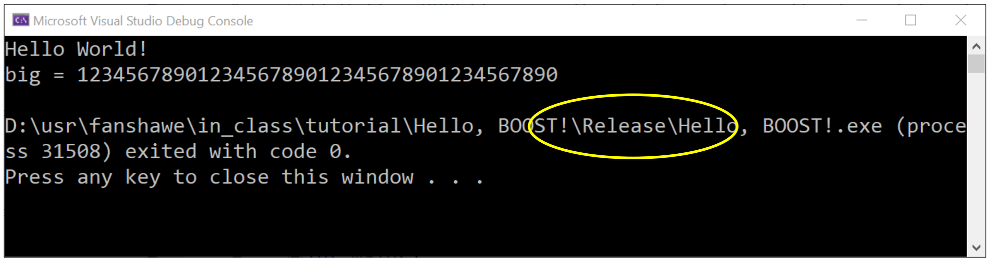
You should see…

And note that the errors have cleared and



```
// Hello, BOOST!.cpp
  execution begins an
//

#include <iostream>
using namespace std;
#include <boost/multi
using namespace boost

int main()
{
    cout << "Hello Wo
    cpp_int big("1234
    cout << "big = "
}
```

Verify the release mode builds



```
Hello World!
big = 1234567890123456789012345678901234567890

D:\usr\fanshawe\in_class\tutorial\Hello, BOOST!\Release\Hello, BOOST!.exe (proce
ss 31508) exited with code 0.
Press any key to close this window . . .
```

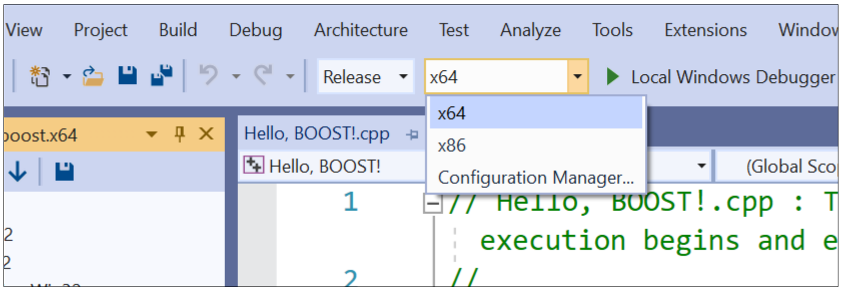
Configure for 64-bit builds

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:

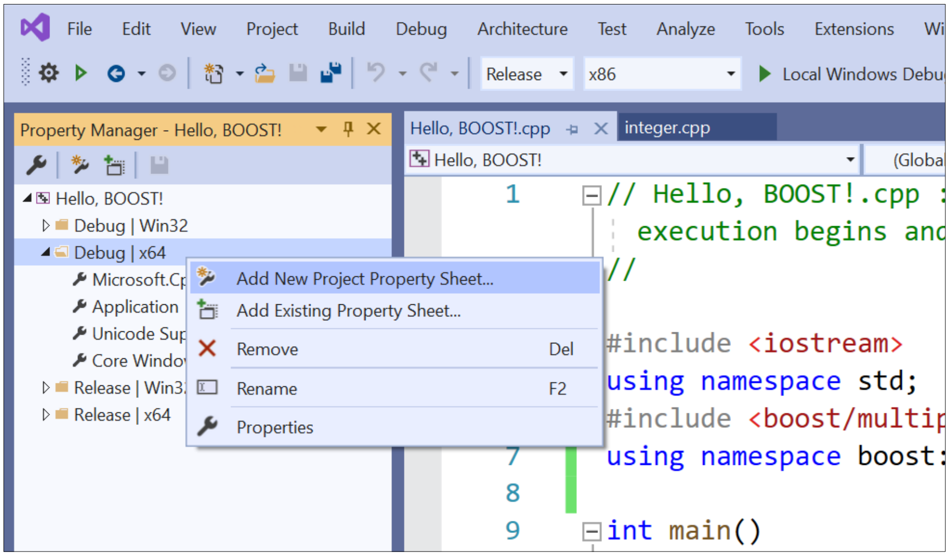
- Create a property sheet in the boost folder
- Set the paths: same include, 64-bit binaries
- Add to both configurations.

Switch to the 64-bit (x64) build.



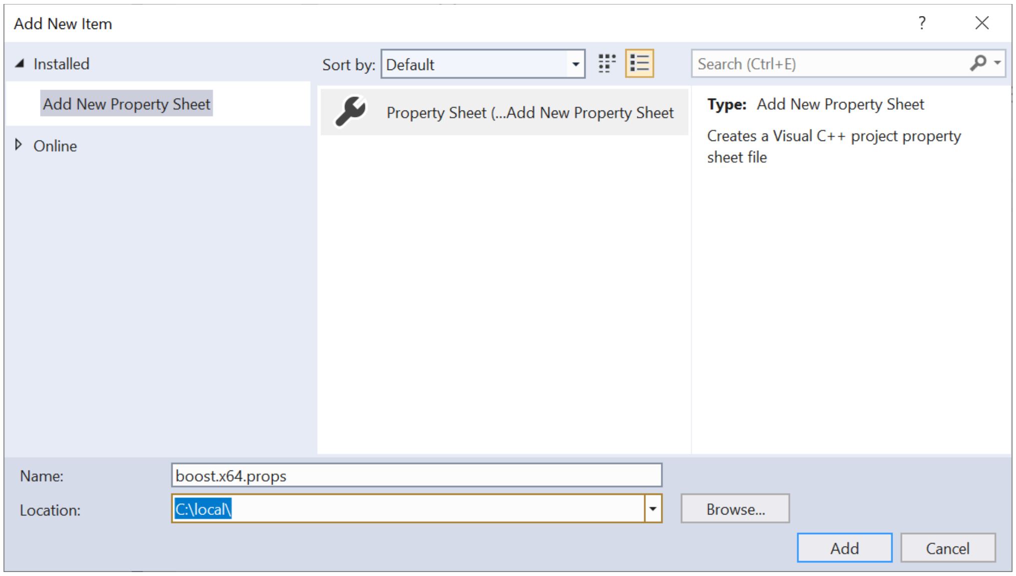
```
// Hello, BOOST!.cpp : T
   execution begins and e
//
```

Add a new project property sheet to Debug | x64



```
// Hello, BOOST!.cpp :
   execution begins and
//

#include <iostream>
using namespace std;
#include <boost/multip
using namespace boost:

int main()
```

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



Type: Add New Property Sheet

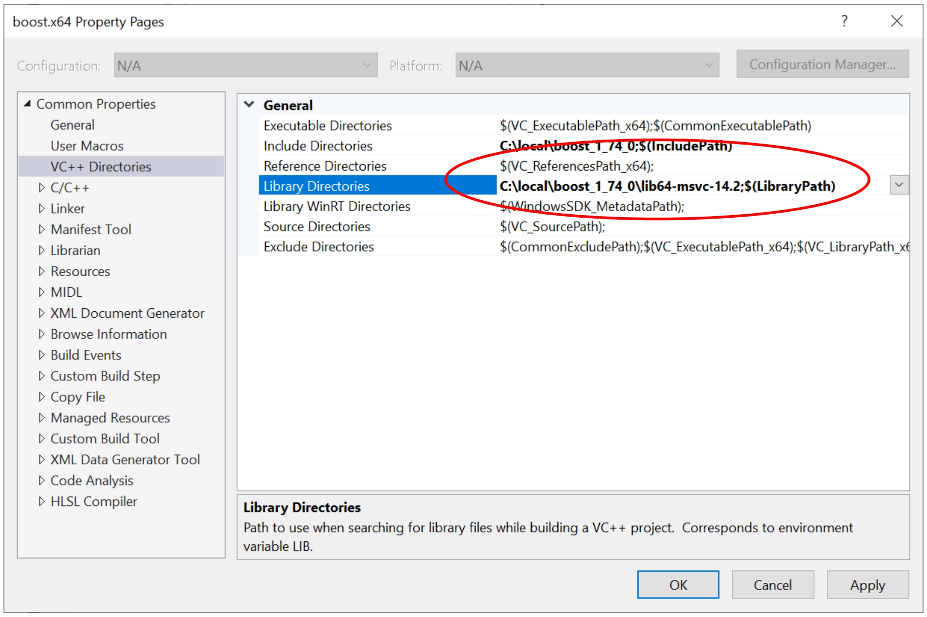
Creates a Visual C++ project property sheet file

Name: boost.x64.props

Location: C:\local\

Configure the paths.

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



boost.x64 Property Pages

Configuration: N/A Platform: N/A

General

Executable Directories	$(VC_ExecutablePath_x64);$(CommonExecutablePath)
Include Directories	**C:\local\boost_1_74_0;$(IncludePath)**
Reference Directories	$(VC_ReferencesPath_x64);
Library Directories	**C:\local\boost_1_74_0\lib64-msvc-14.2;$(LibraryPath)**
Library WinRT Directories	$(WindowsSDK_MetadataPath);
Source Directories	$(VC_SourcePath);
Exclude Directories	$(CommonExcludePath);$(VC_ExecutablePath_x64);$(VC_LibraryPath_x6

Library Directories
Path to use when searching for library files while building a VC++ project. Corresponds to environment variable LIB.

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.

Document History

Version	Date	Activity
0.0.0	2009	Document created
1.0.0	2020-09-09	Updated for Visual Studio Community 2019
1.1.0	2023-10-29	Updated for Visual Studio Community 2022