GED Practical Course



computer graphics & visualization

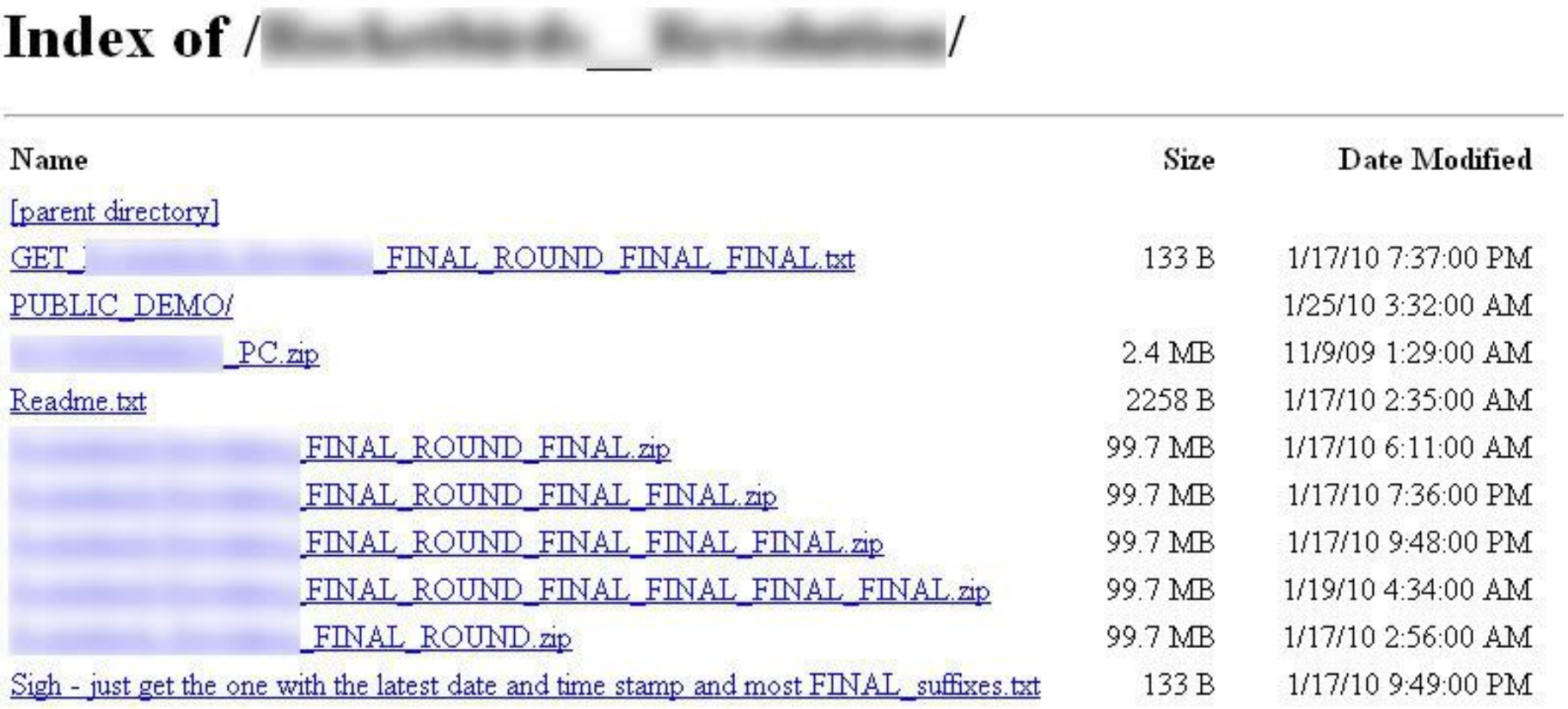
Incoming

Incoming: Previous Results

Incoming

- Build a game „from scratch“
  - No existing engine
- Of course we can not write a game engine in one semester…
  - Focus on rendering (DirectX)
  - Also a bit of particle systems, physics, game object management, configurations etc
- But let‘s start slowly

SVN (and other SCM)



Index of /[redacted]/

Name	Size	Date Modified
[parent directory]		
GET_[redacted]_FINAL_ROUND_FINAL_FINAL.txt	133 B	1/17/10 7:37:00 PM
PUBLIC_DEMO/		1/25/10 3:32:00 AM
[redacted]_PC.zip	2.4 MB	11/9/09 1:29:00 AM
Readme.txt	2258 B	1/17/10 2:35:00 AM
[redacted]_FINAL_ROUND_FINAL.zip	99.7 MB	1/17/10 6:11:00 AM
[redacted]_FINAL_ROUND_FINAL_FINAL.zip	99.7 MB	1/17/10 7:36:00 PM
[redacted]_FINAL_ROUND_FINAL_FINAL_FINAL.zip	99.7 MB	1/17/10 9:48:00 PM
[redacted]_FINAL_ROUND_FINAL_FINAL_FINAL_FINAL.zip	99.7 MB	1/19/10 4:34:00 AM
[redacted]_FINAL_ROUND.zip	99.7 MB	1/17/10 2:56:00 AM
Sigh - just get the one with the latest date and time stamp and most FINAL_suffixes.txt	133 B	1/17/10 9:49:00 PM

SVN (and other SCM)

- Software Configuration Management
- Tools to solve problems like:
  - Multiple developers working on the same project
  - Tracking changes

- **Each** project with more than a single developer needs an SCM system
- Benefits for a single developer, too: Rollback, Branches, etc.

SVN

- Centralized: one server, multiple clients
- Each change must be „commited“ to the server
  - Each „Commit“ increases the revision number
- All changes are „atomic“: repository always consistent
- Each revision can be restored
  - Tracking changes
  - Reverting errors

SVN

- Sourcecode history is stored on the server

- You work on a „working copy“
  - Basically a copy of a revision
  - May contain local changes
  - „Per file“ basis
    - So only a part of your working copy might be up-to-date if you choose so

SVN

- *Every* file of the project needs to be in your SVN
  - Everything needed to build the program
  - You will work on multiple PCs!
  - Check regularly if this is the case

- Do *not* add files created from your code to SVN
  - *.exe, *.dll
  - Automatically created and temporary files
    - *.opensln, *.sdf, etc…
    - See the first assignment for details

SVN Commands

- `svn co` (checkout): Create a working copy of an SVN folder
- `svn add`: Add files to the SVN
- `svn ci` (commit): Commit changes of your working copy to the server
- `svn up` (update): Update your local copy with changes from the server
- `svn log`: Show changes for a file / folder

SVN

- „Commit early, commit often“
  - So you can also trace small changes with great impact…
- `update` before you start working
- `update` before and after a commit
  - Reduces conflicts

- Ignore list: Add `Debug`, `Release`, `.ncb`, `.suo`, etc.

- **Always** include a **meaningfull** Commit-Message
  - Helps you track down changes

SVN

- 2 Repositories:
  - https://svnge.in.tum.de/SS15/external
    - Contains Meshes, Textures, Examples, Libraries, Assignments, Slides, etc
    - Regular updates!
  - https://svnge.in.tum.de/SS15/<username>
    - Your source code

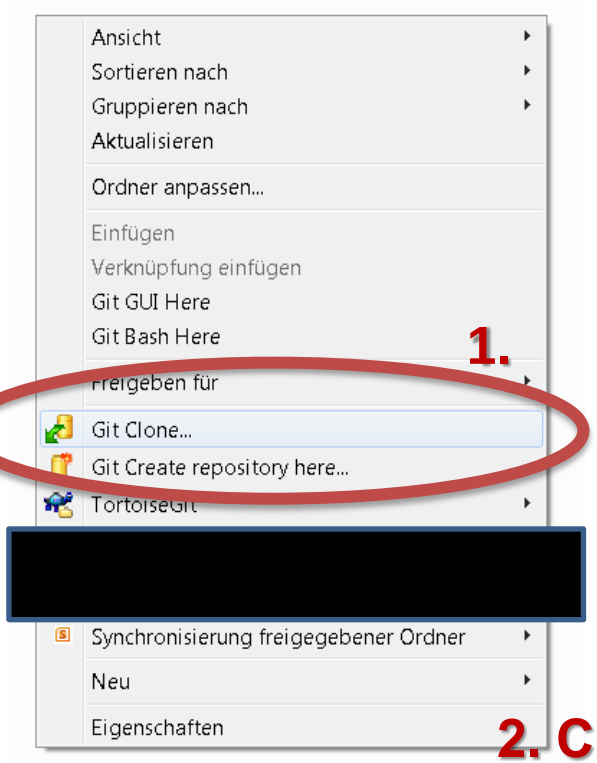
- Help!
  - SVN: http://svnbook.red-bean.com/index.en.html
  - TortoiseSVN: http://tortoisesvn.net/support.html

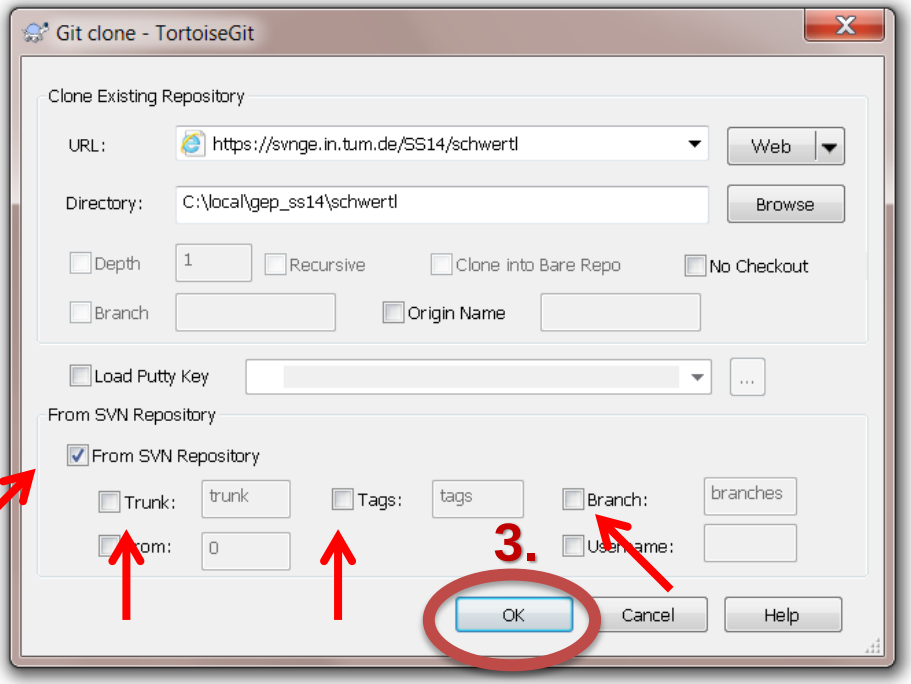
SVN: GED Rooms

- You can checkout your repository, to a folder of your choice, say `.\gep\<username>`
  - Do not forget to commit before logging out!

- Checkout `external` to `.\gep\external`
  - Update before you start working!

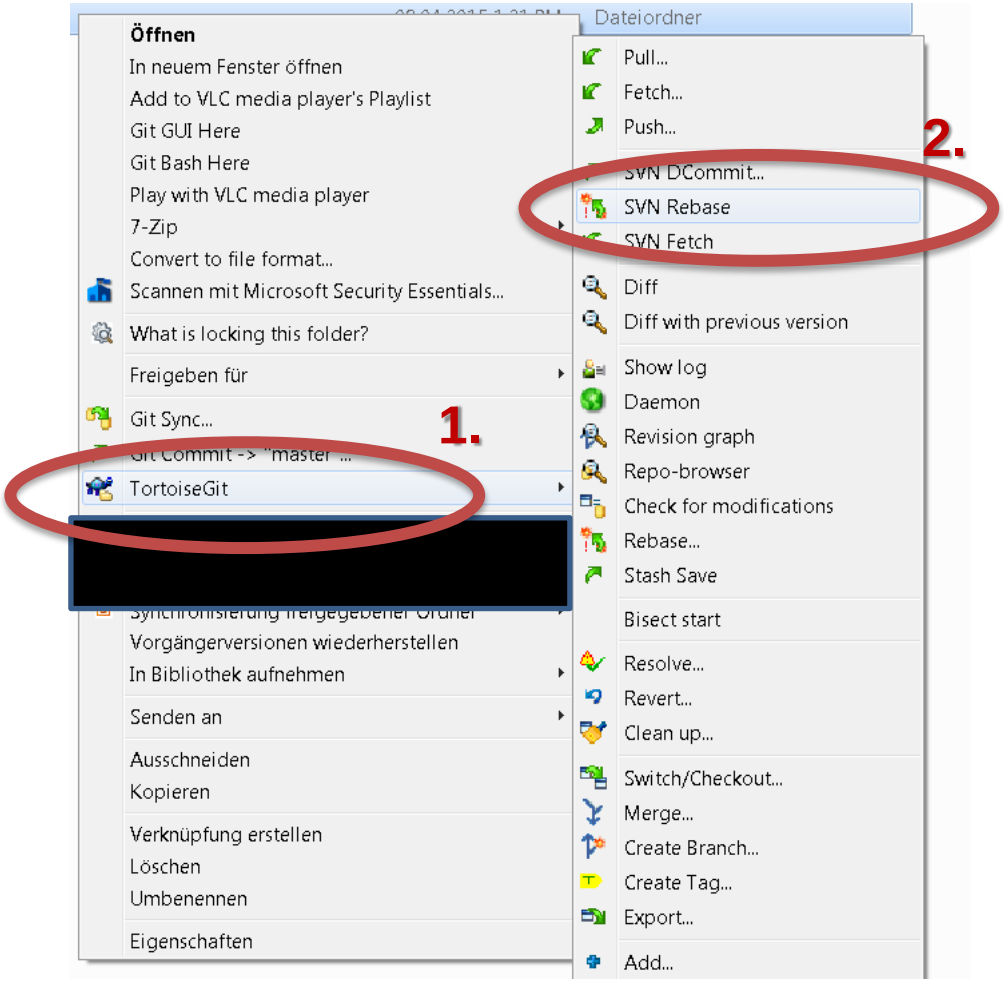
- Checkout with TortoiseGit





2. Check "From SVN Repository" and uncheck "Trunk:", "Tags:" and "Branch:"

- Update with TortoiseGit

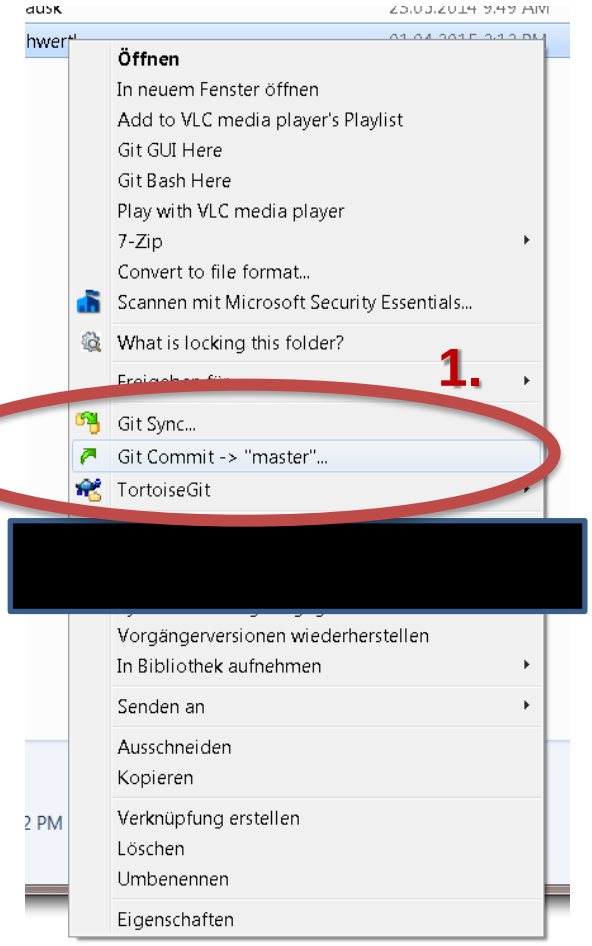


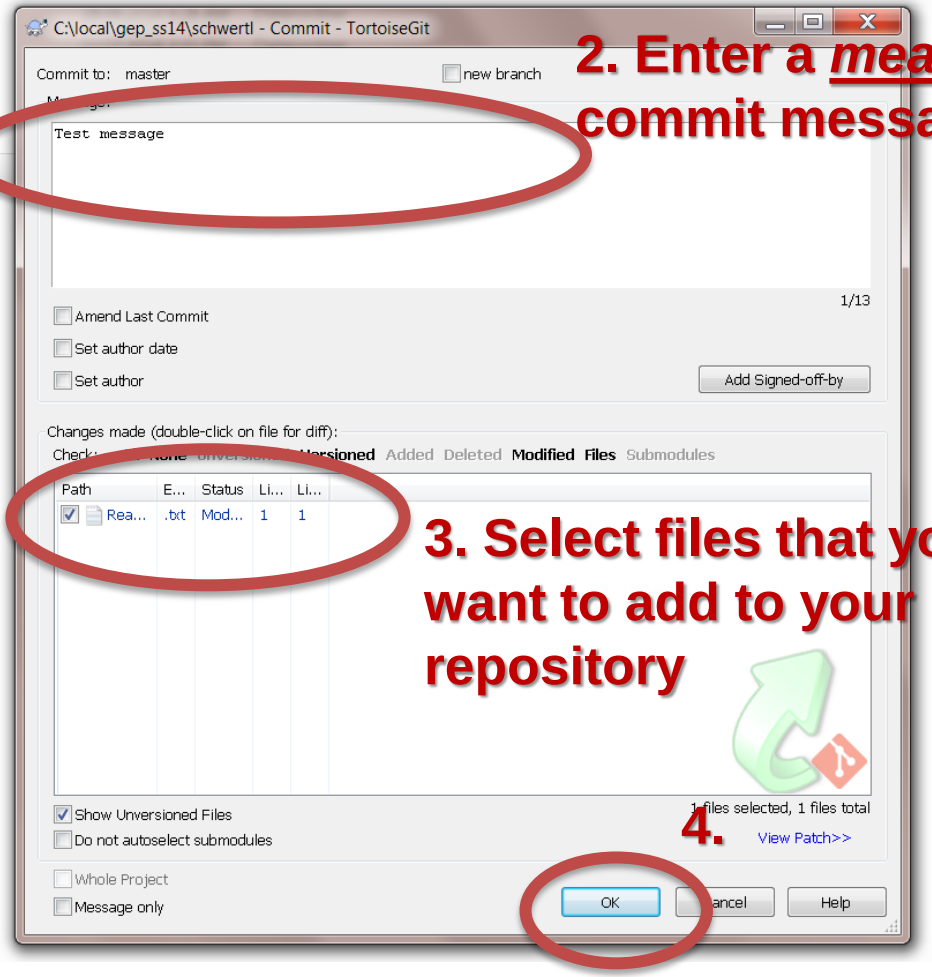
SVN

- Checkout `external` and `<username>` to the same folder!

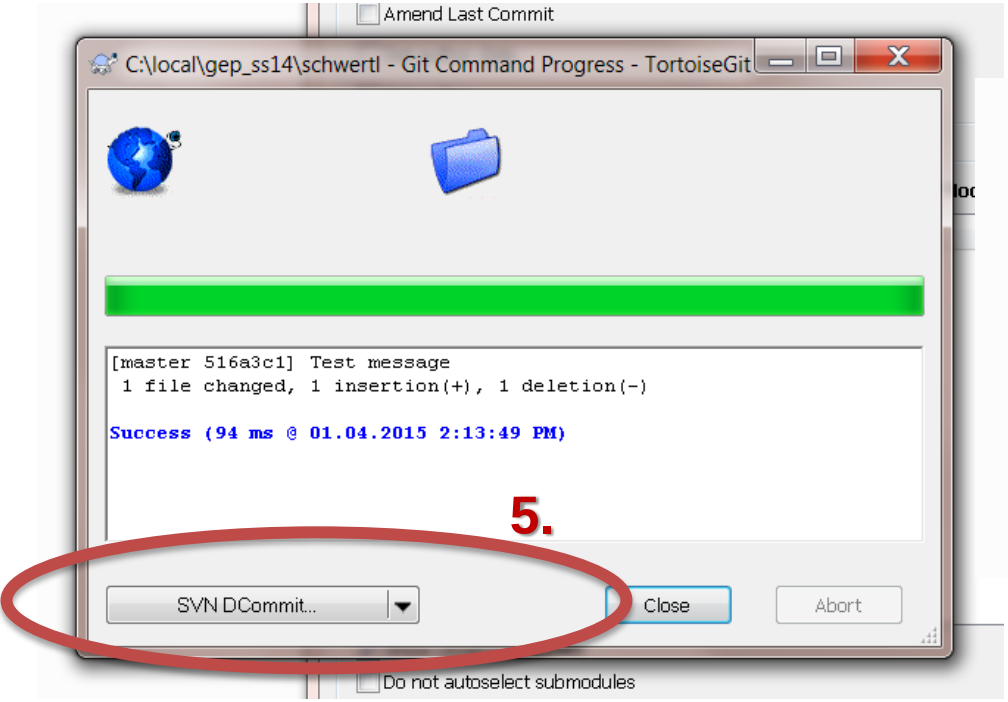
- Only work in your folder, don‘t change anything in external

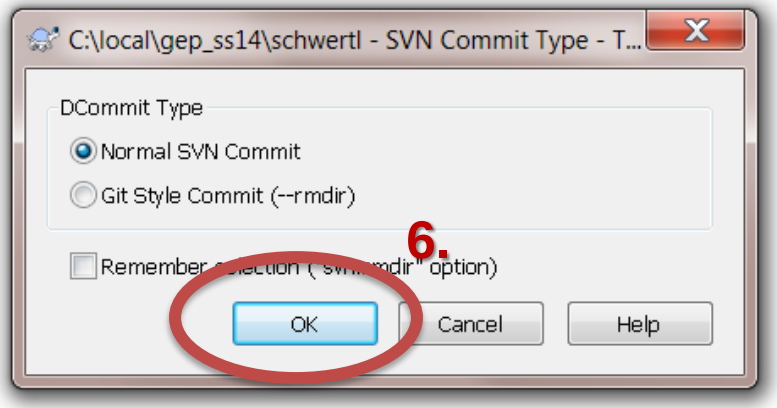
- Commit with TortoiseGit





- Commit with TortoiseGit (continued)

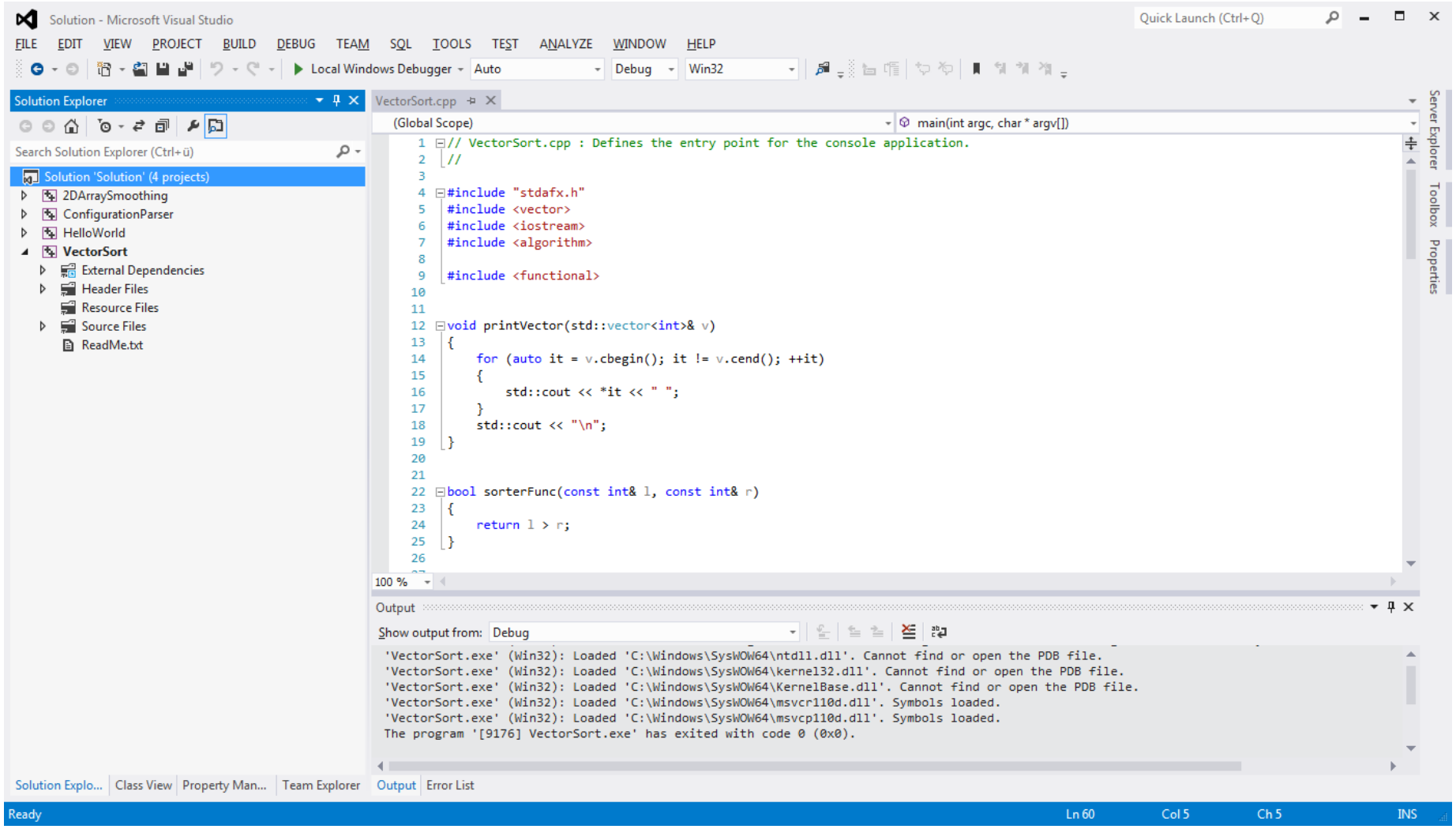




Visual Studio

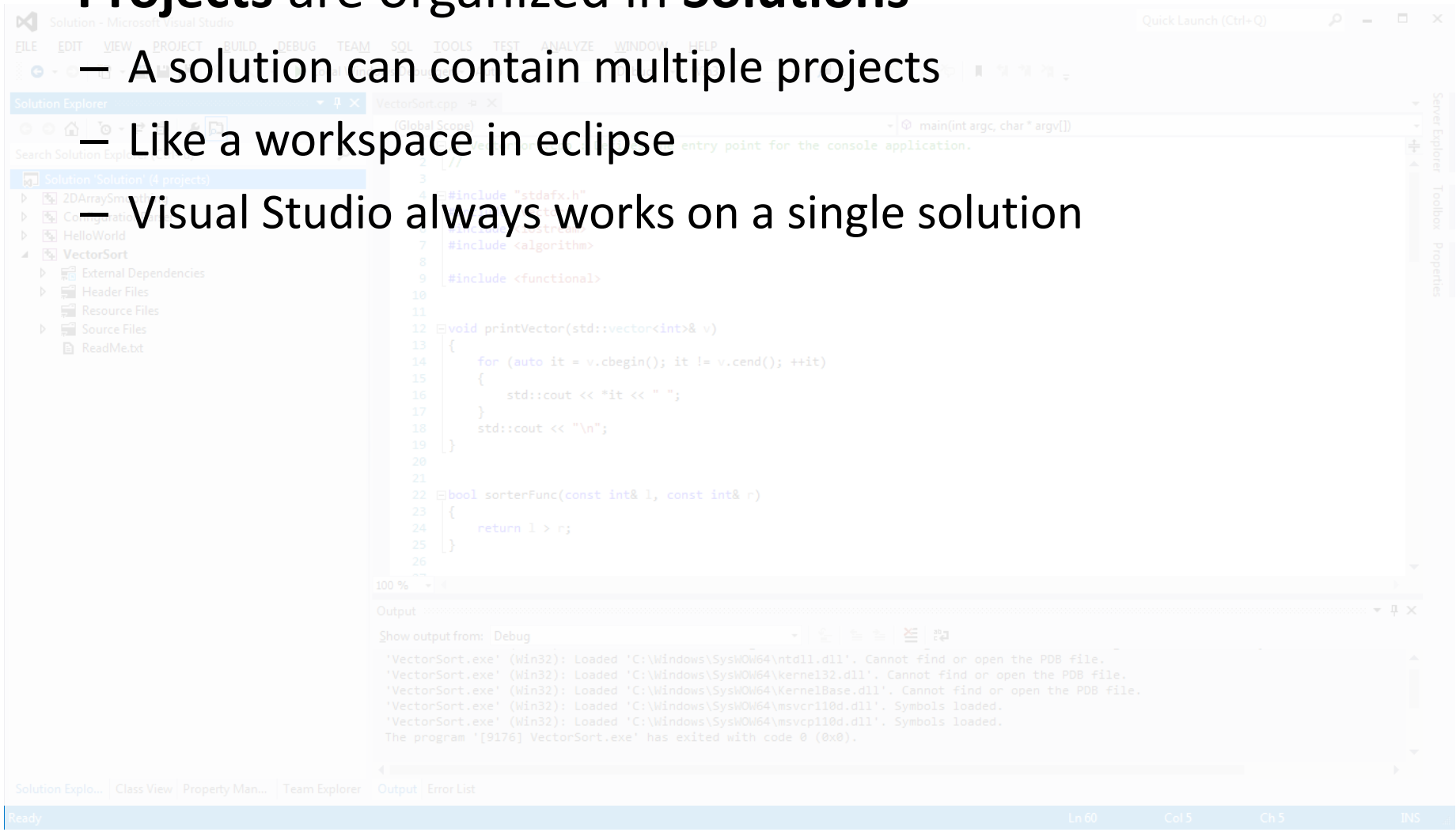
- Microsofts IDE for C++, C#, VB, F#...
  - Not just a compiler
  - Editor w/ syntax highlighting, code completion etc
  - Debugger
  - Several other tools you won‘t need

Visual Studio

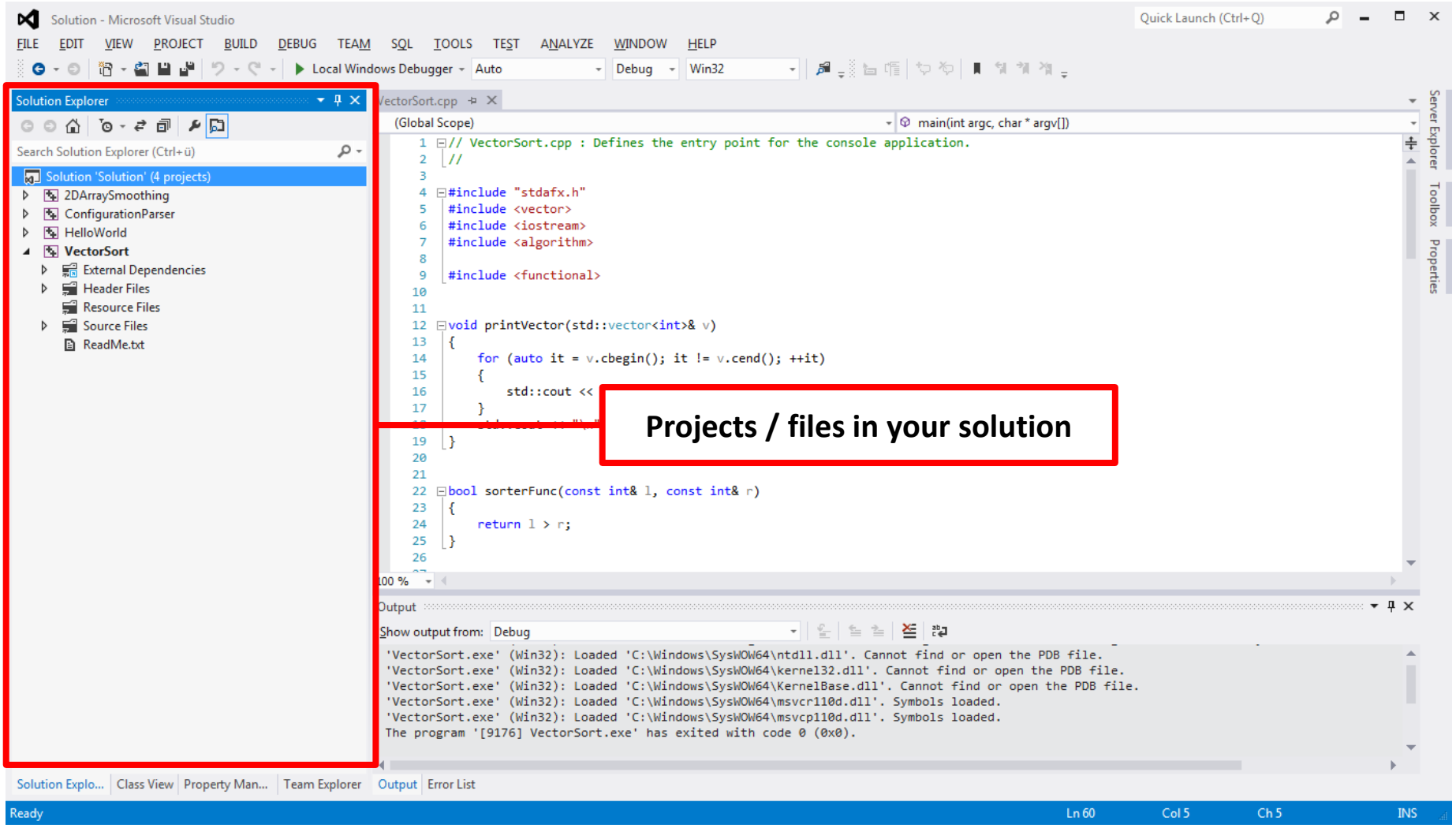


Visual Studio

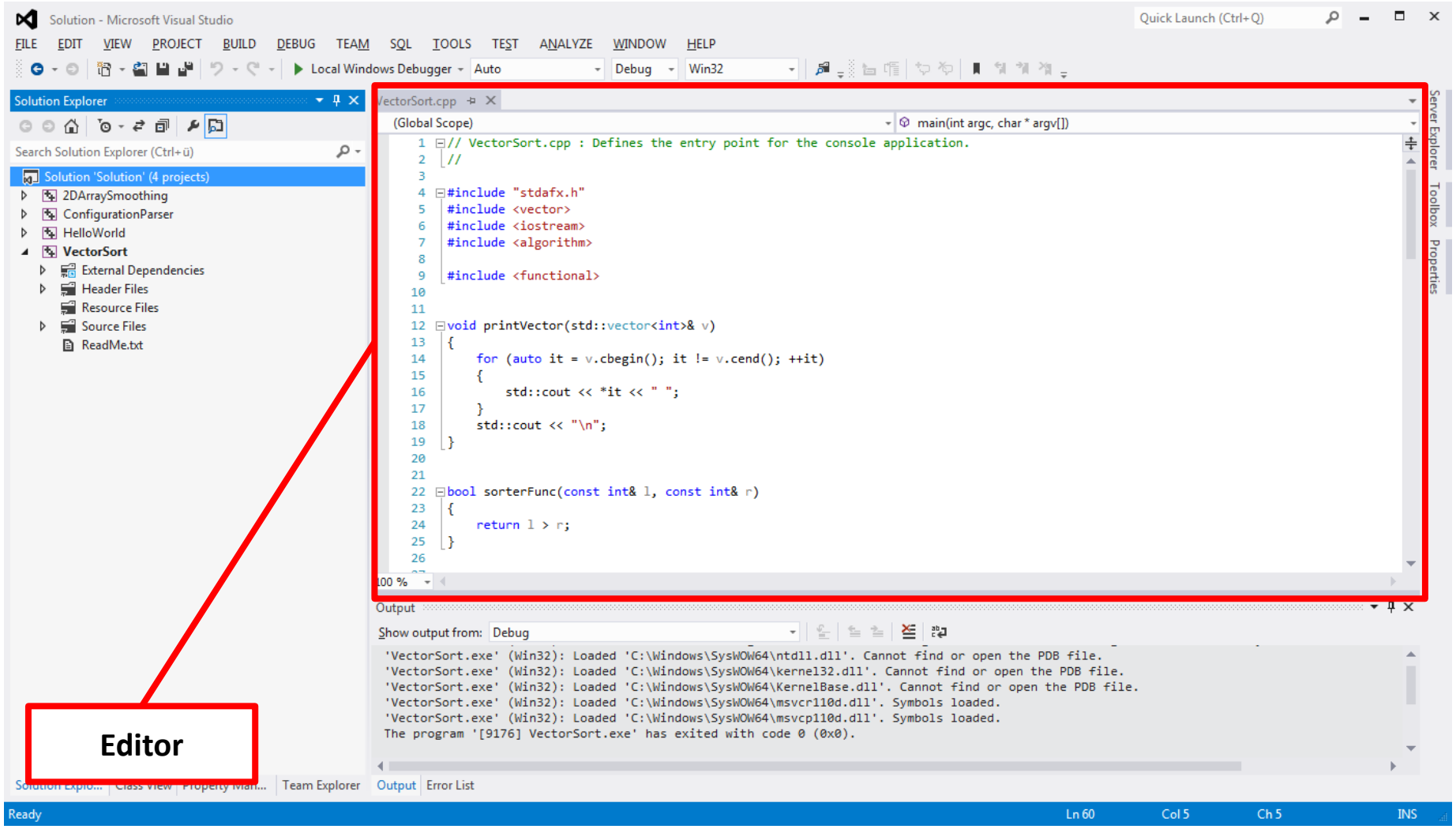
- **Projects** are organized in **Solutions**
  - A solution can contain multiple projects
  - Like a workspace in eclipse
  - Visual Studio always works on a single solution



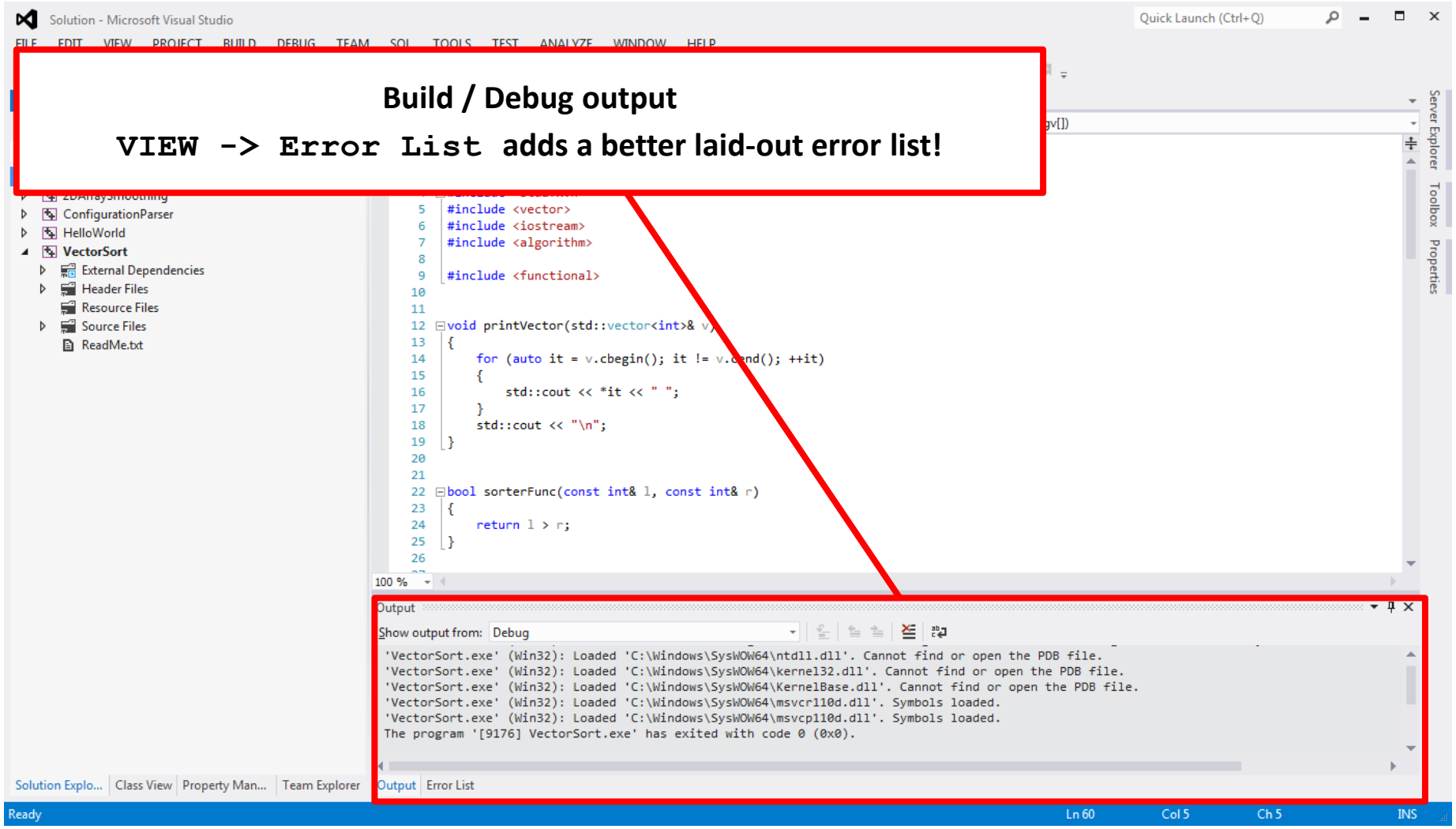
Visual Studio



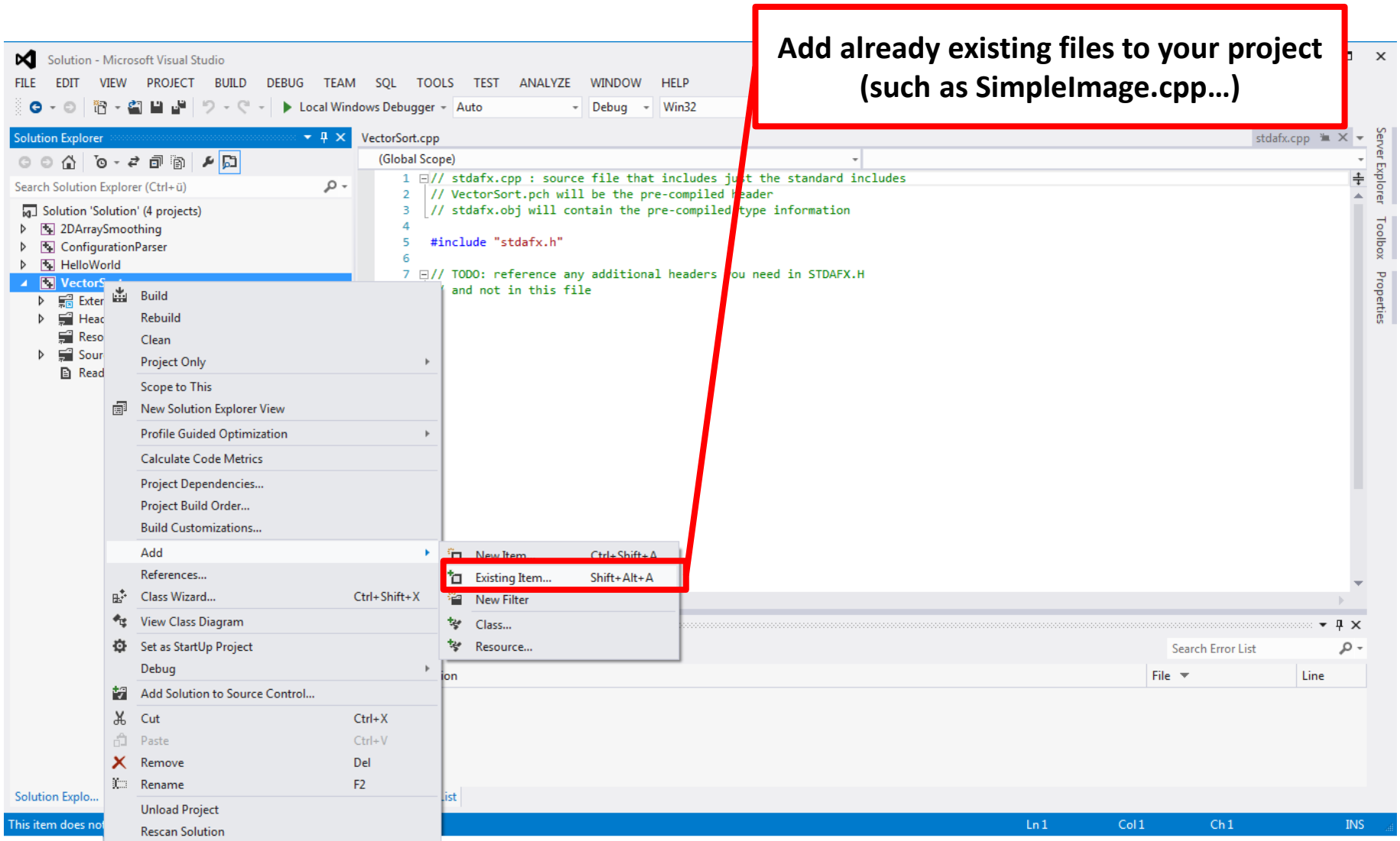
Visual Studio



Visual Studio

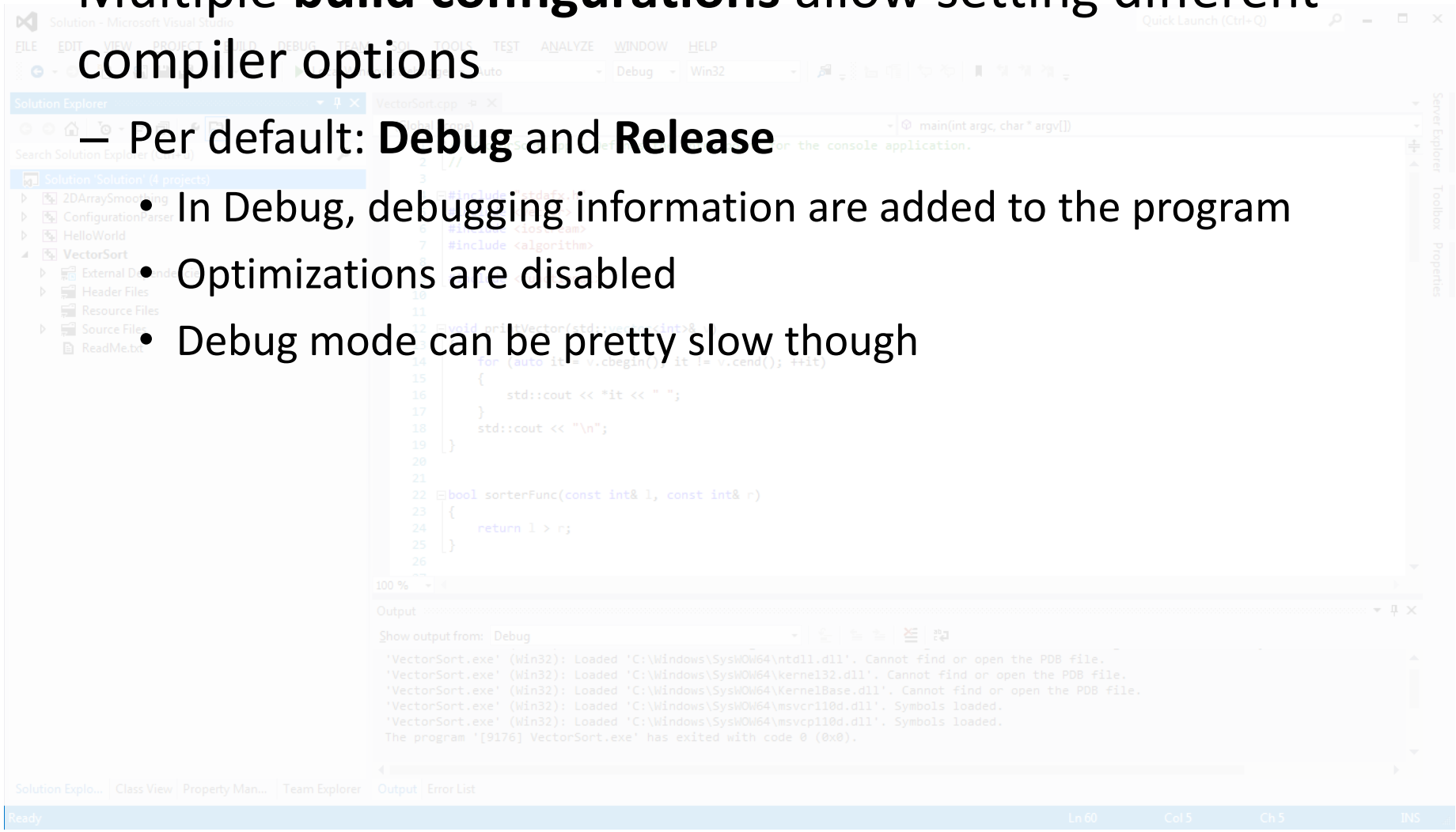


Visual Studio

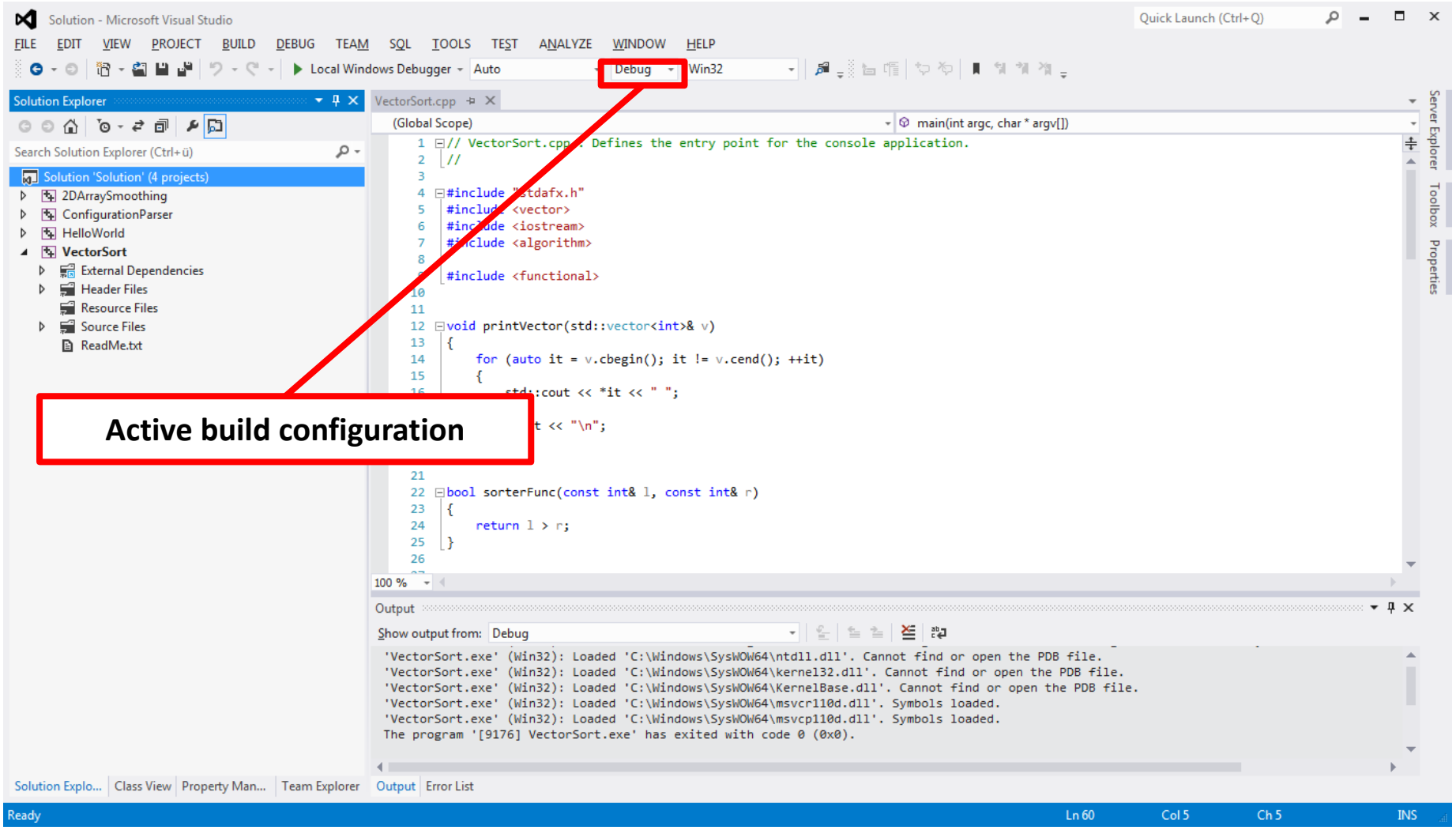


Visual Studio

- Multiple **build configurations** allow setting different compiler options
  - Per default: **Debug** and **Release**
    - In Debug, debugging information are added to the program
    - Optimizations are disabled
    - Debug mode can be pretty slow though



Visual Studio

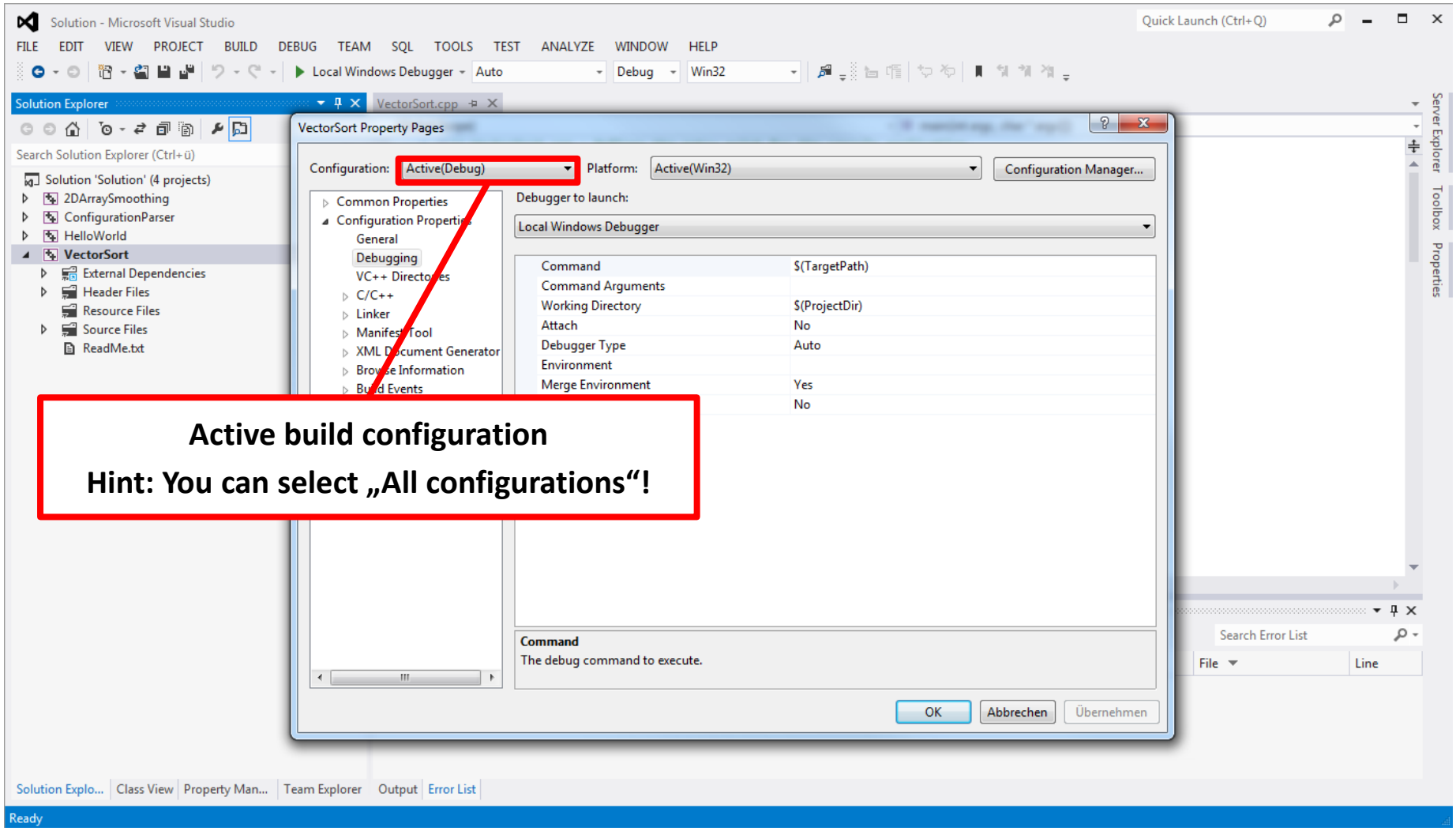


Visual Studio

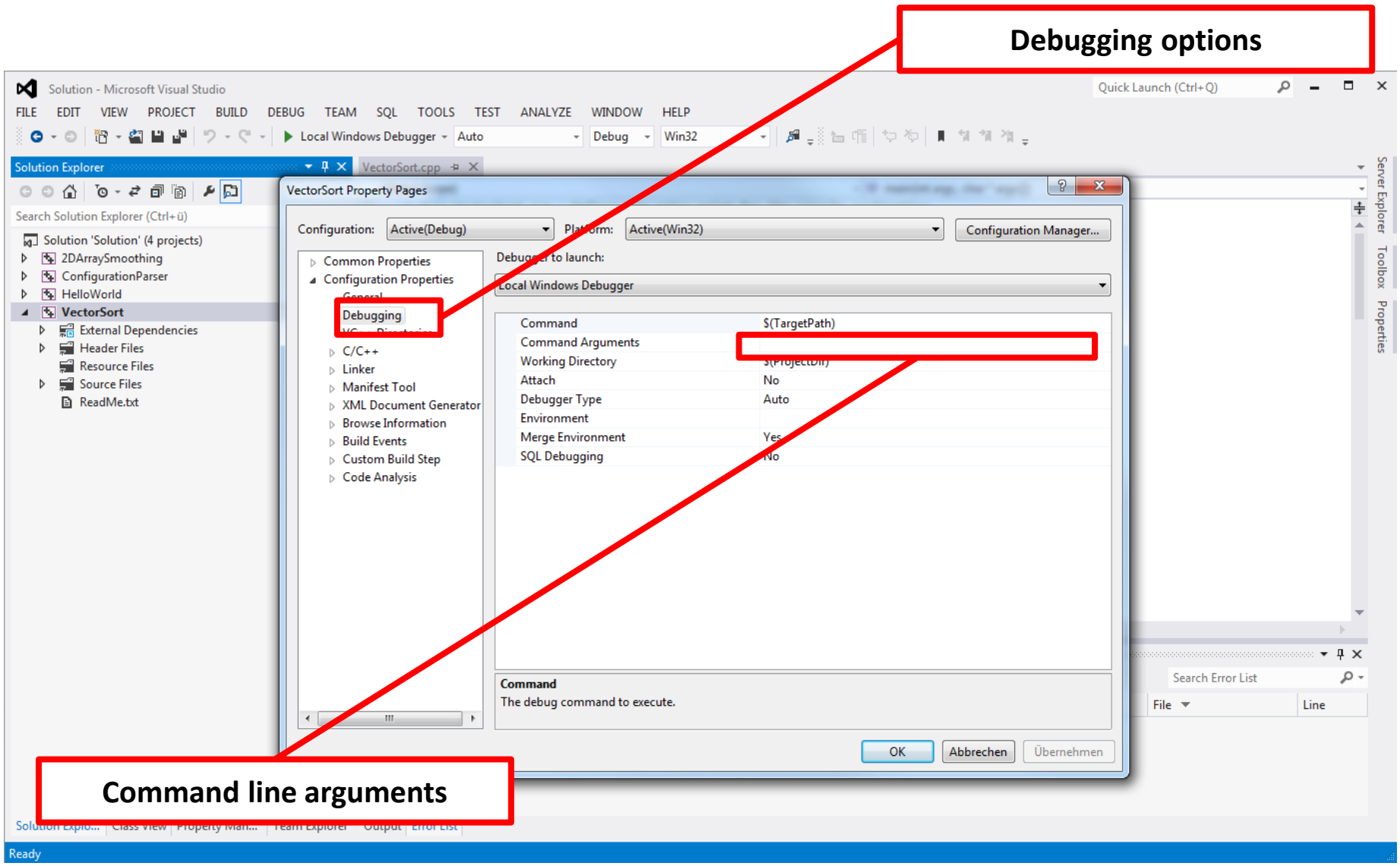
- PROJECT -> Properties shows the settings of the currently selected project
  - Can be altered for each build configuration
  - Change compiler / linker options
  - Set debugging options
  - Add search directories for header files and libraries
  - Be sure to use **relative** path names if possible!

```
..\..\..\external\SimpleImage\
```

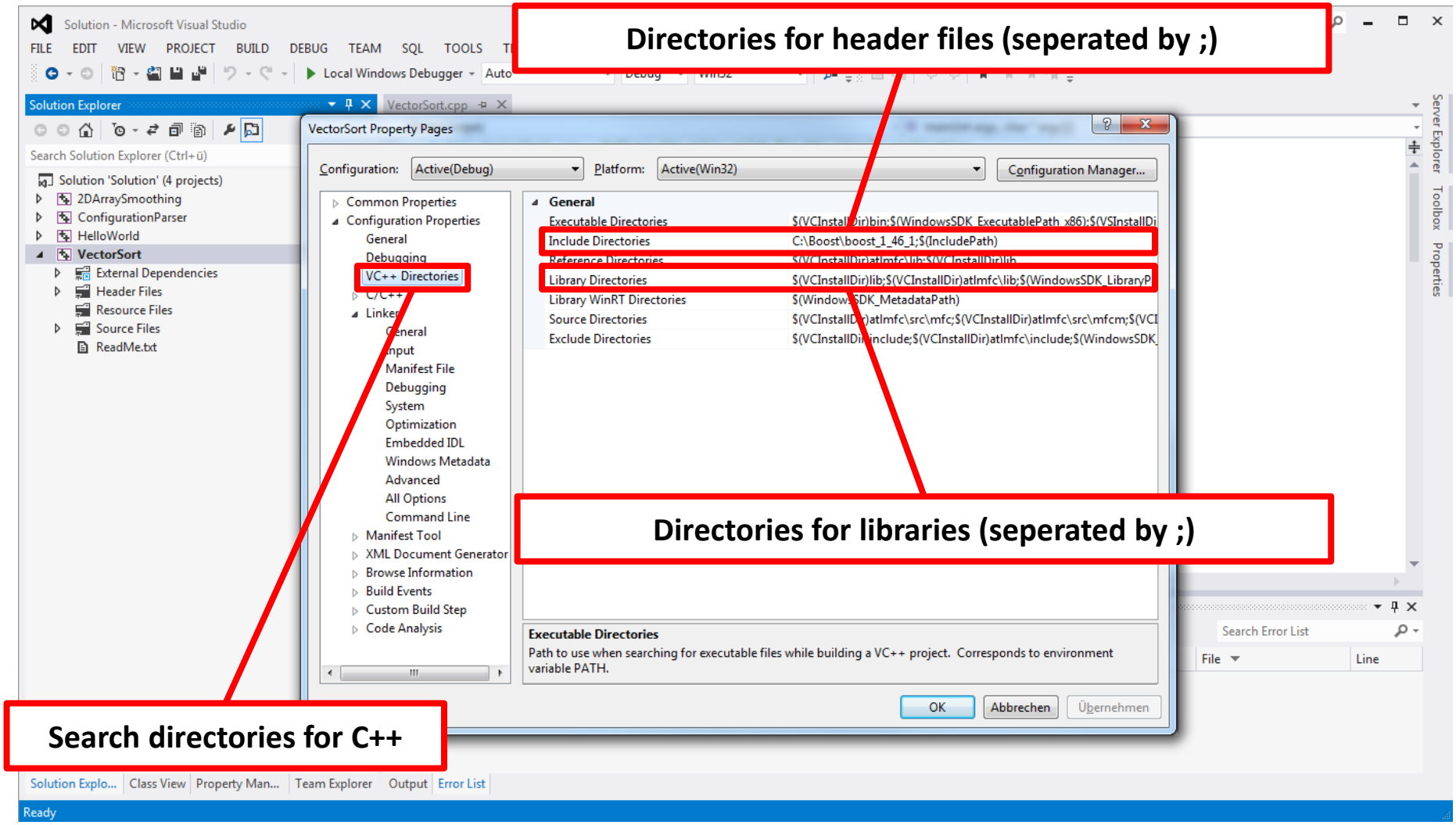
Visual Studio



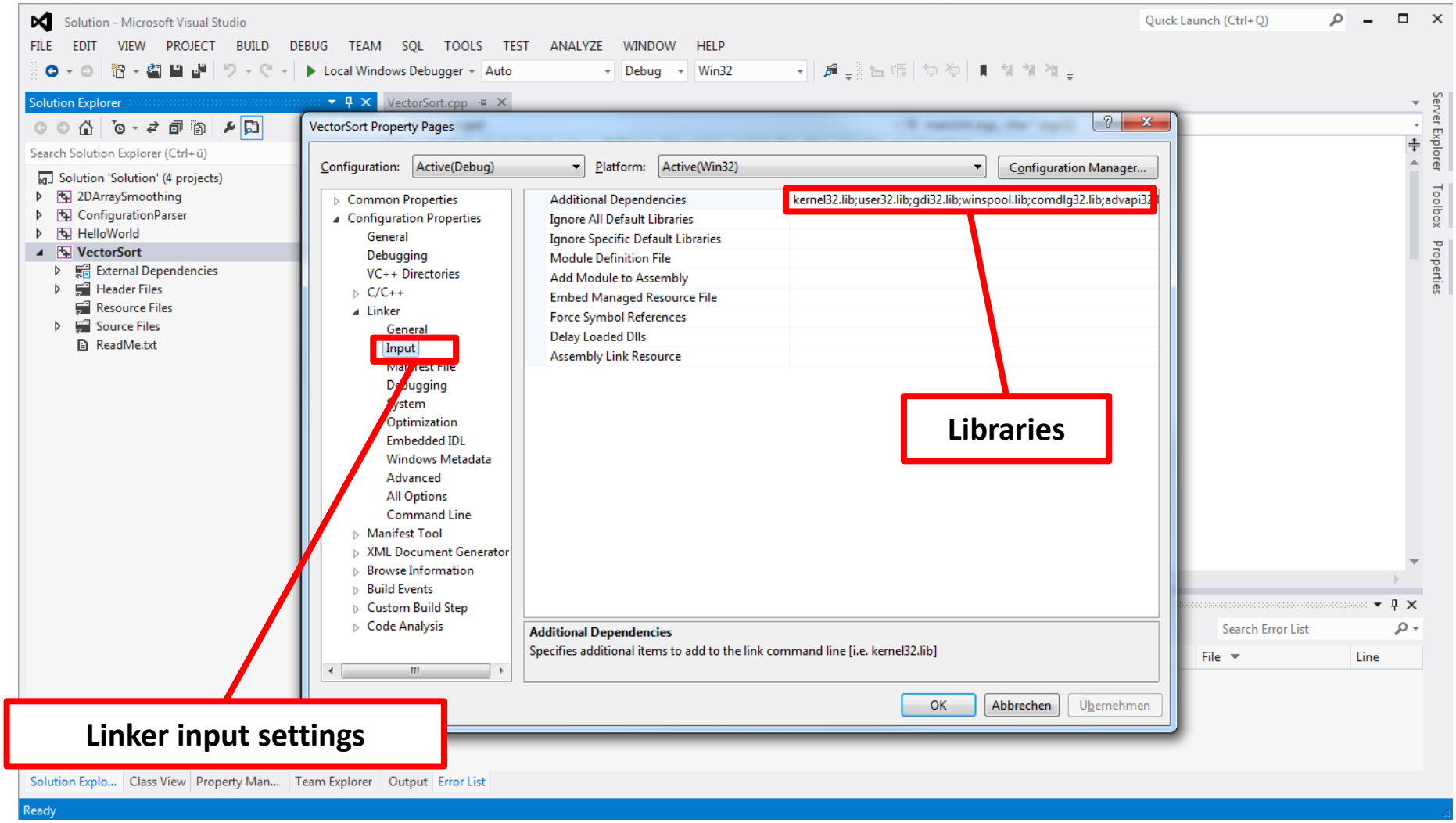
Visual Studio



Visual Studio



Visual Studio



Questions?

You can also use our Q&A forum for further questions.