Software Testing

Version Control Systems

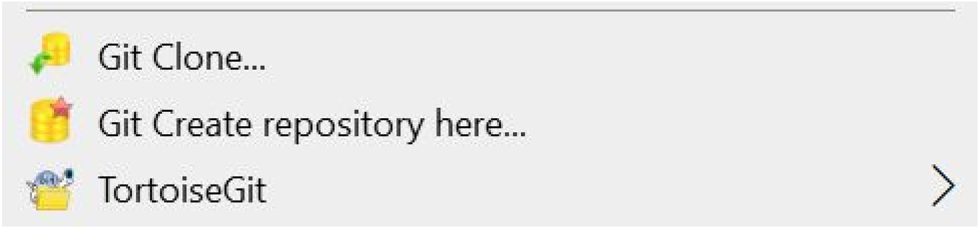
Version Control

- Version control systems are
  - repositories for software that are able to maintain multiple versions of the software.
  - This means it is able to save the previous version as well as the new version.
  - The advantage of this is that if you make a mistake and you need to revert to a previous version, this can easily be done.
- Version control can be run
  - Locally on your computer
  - Remotely on a network server
- Git is one of the most popular version control systems today.
- Seneca runs its own server at:
  - https://github.senecacollege.ca
  - You must use the VPN to access this

Git Clients

- A Git client is a program which communicates with either a local or remote Git repository.
- The Git client is able to
  - creating a clone of the repository,
  - add new files to the repository, and
  - retrieve information about the repository.
- There are many different get clients available.
  - TortoiseGit is integrated into the Windows file browser

TortoiseGit Menus

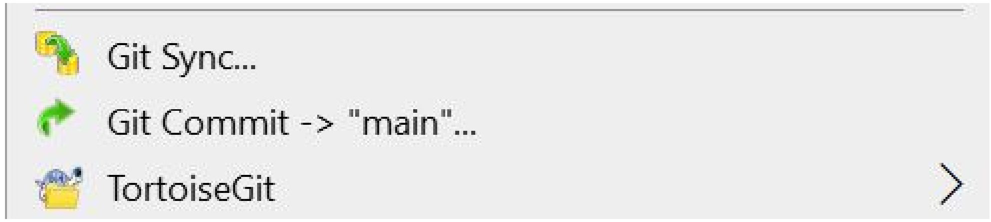


Git Clone...

Git Create repository here...

TortoiseGit

Non-repository



Git Sync...

Git Commit -> "main"...

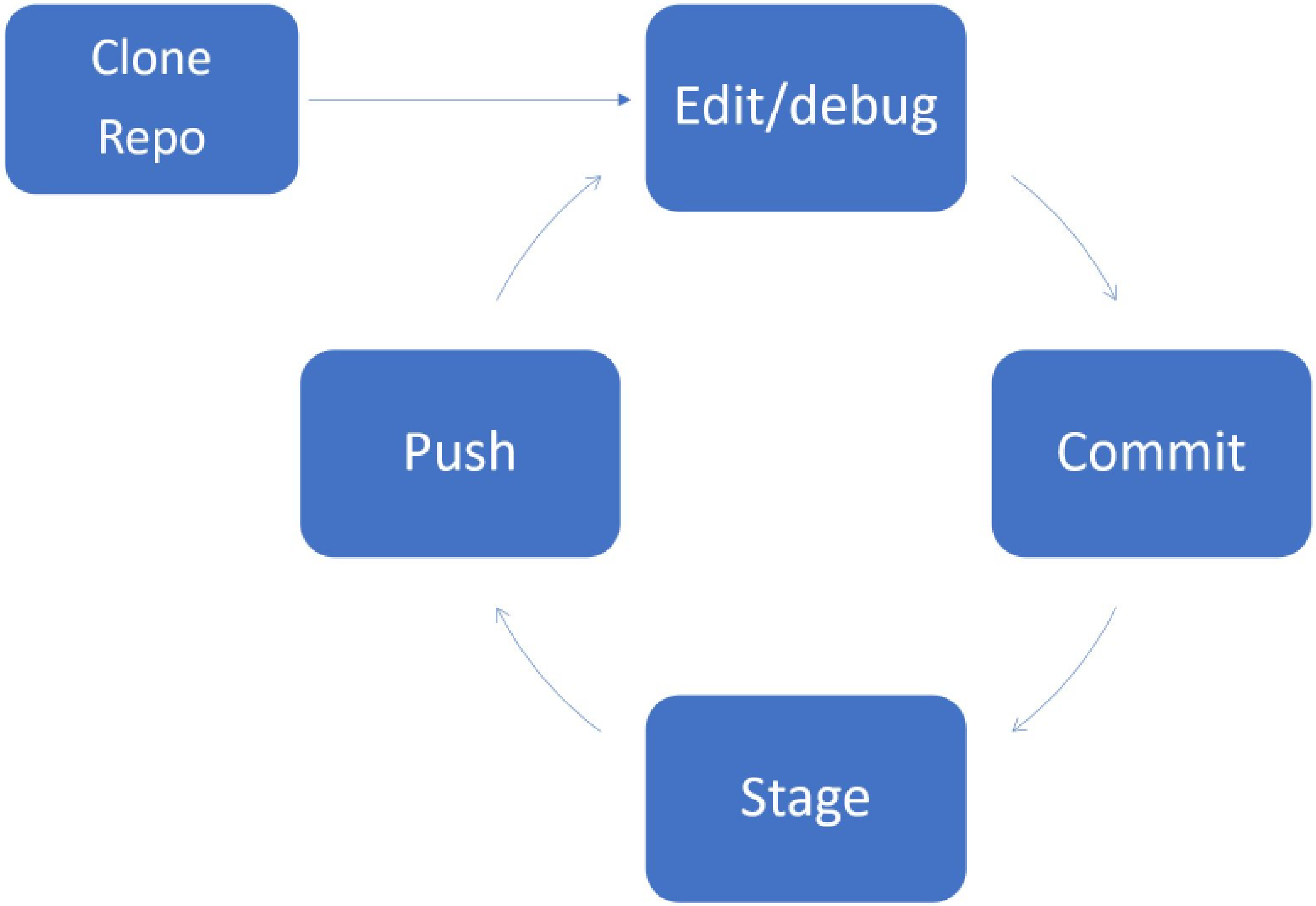
TortoiseGit

Repository

Working Copies

- You clone the remote repository to your computer
- This creates a local copy of the repository
- You make changes to the local repository
- You commit the changes to the local repository
- Stage the files you want to be in the next push
- You push the committed changes to the remote repository so others can share your work

Working Copies



Clone Repo

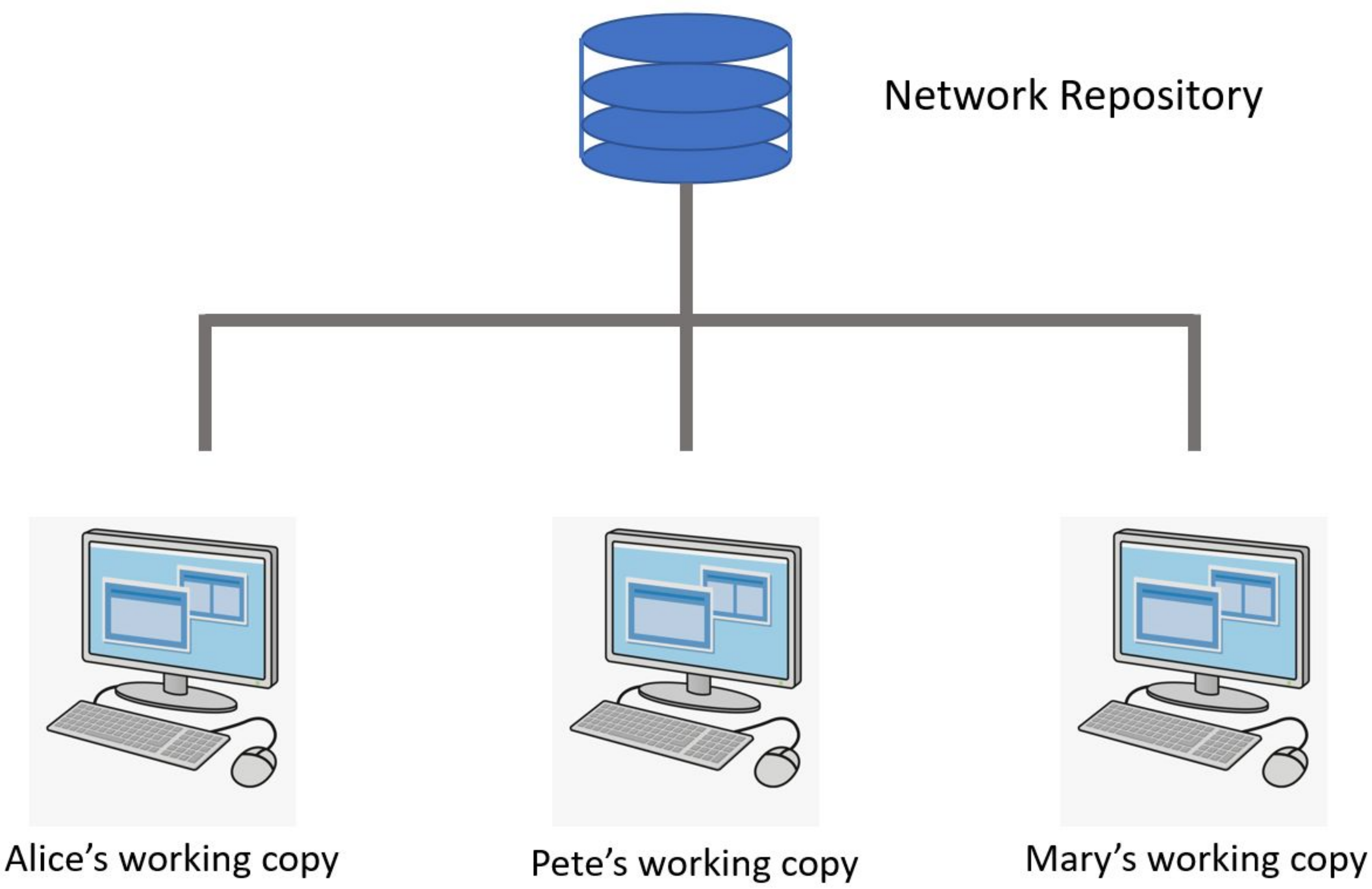
Edit/debug

Commit

Stage

Push

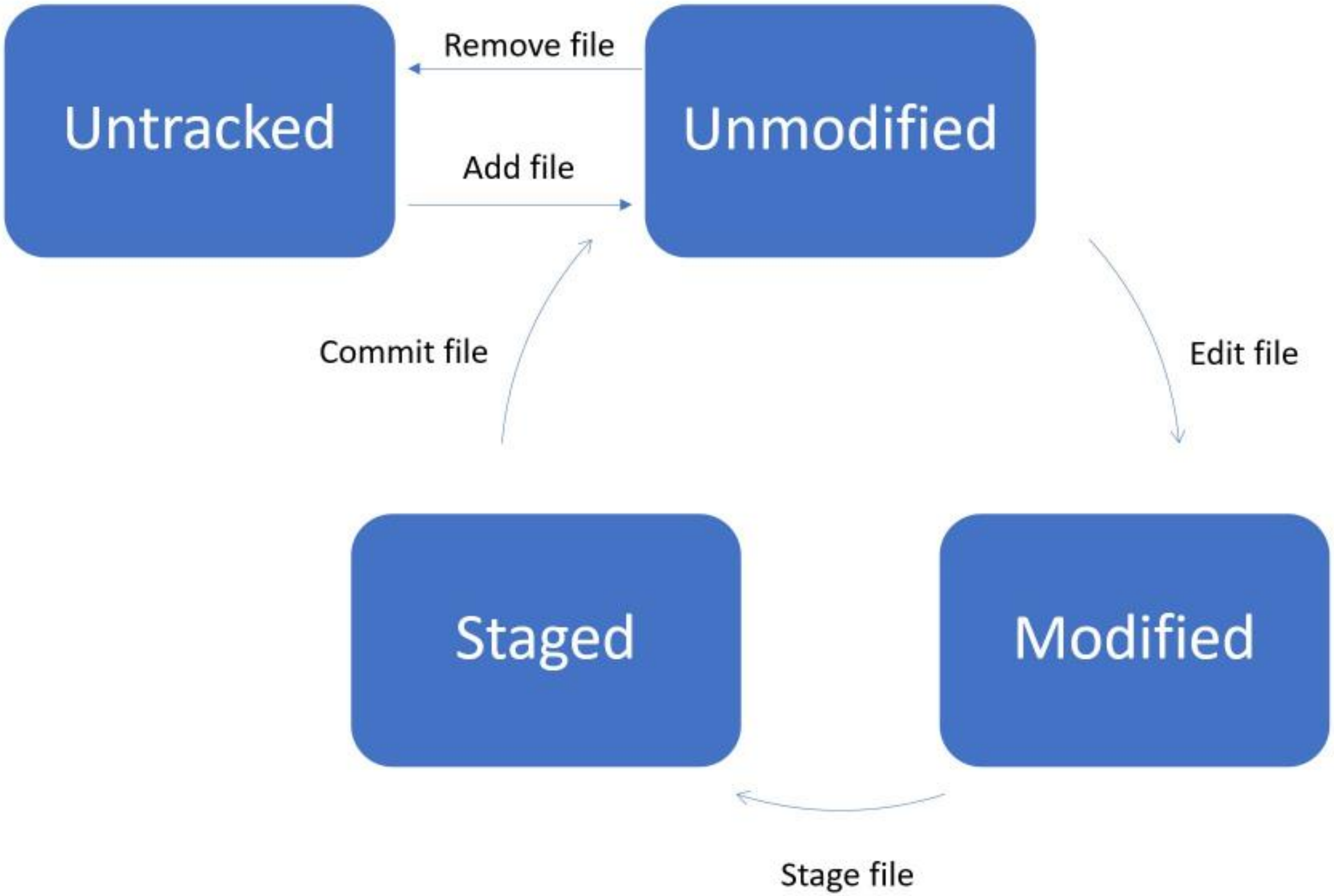
Working Copies



File Status

- Git only versions the files you ask it to
  - Projects have many intermediate file like object and executable files which can be rebuilt
- You must add a file to Git to have it tracked or versioned
- Once tracked, it tells you if the file has been modified
- You can then stage the file and commit it

File Status



Untracked

Remove file

Add file

Unmodified

Commit file

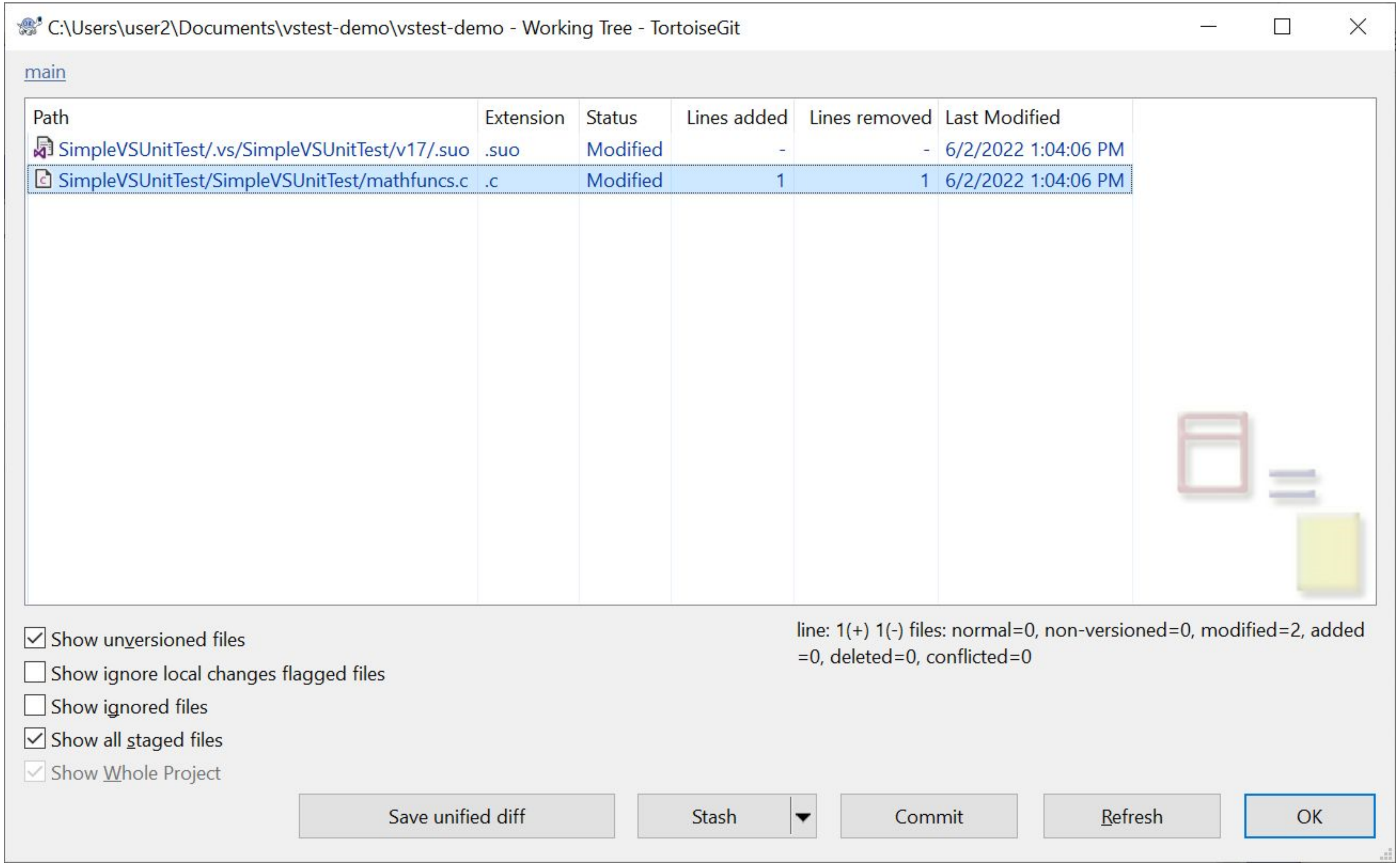
Edit file

Staged

Modified

Stage file

Tortoise: Check for Modifications



C:\Users\user2\Documents\vstest-demo\vstest-demo - Working Tree - TortoiseGit

main

Path	Extension	Status	Lines added	Lines removed	Last Modified
SimpleVSUnitTest/.vs/SimpleVSUnitTest/v17/.suo	.suo	Modified	-	-	6/2/2022 1:04:06 PM
SimpleVSUnitTest/SimpleVSUnitTest/mathfuncs.c	.c	Modified	1	1	6/2/2022 1:04:06 PM

line: 1(+) 1(-) files: normal=0, non-versioned=0, modified=2, added =0, deleted=0, conflicted=0

☑ Show unversioned files

□ Show ignore local changes flagged files

□ Show ignored files

☑ Show all staged files

☑ Show Whole Project

Ignoring Files

- You want Git to ignore temporary files
- List patterns for them in a .gitignore file

```
*.vsidx
*.lock
*.ipch
*.testlog
*.tlog
*.log
*.VC.db
*.manifest
*.VC.db-shm
*.VC.db-wal
*.VC.db-opendb
```

Communicating with the Remote Repository

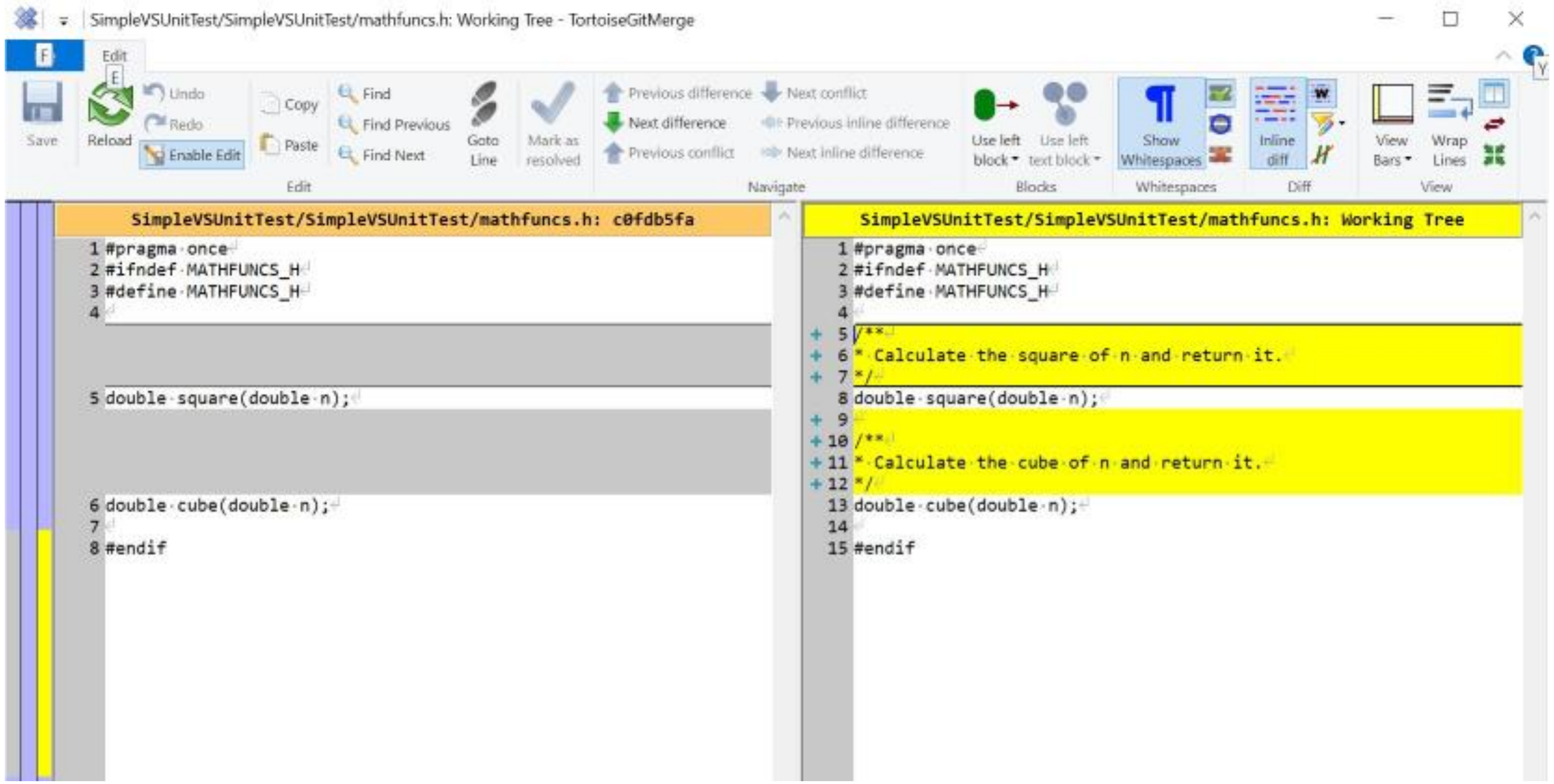
- **Clone** – creates a local copy of the repository
- **Pull** – updates your local copy with changes from the remote
- **Push** – pushes your local changes to the remote

Other Actions

- **Rename** – change the name of a file tracked by Git
- **Delete** – stop a file from being tracked by Git

Viewing Changes

Check for Modifications | Compare with Base



SimpleVSUnitTest/SimpleVSUnitTest/mathfuncs.h: Working Tree - TortoiseGitMerge

SimpleVSUnitTest/SimpleVSUnitTest/mathfuncs.h: c0fdb5fa

```
#pragma once
#ifndef MATHFUNCS_H
#define MATHFUNCS_H

double square(double n);

double cube(double n);

#endif
```

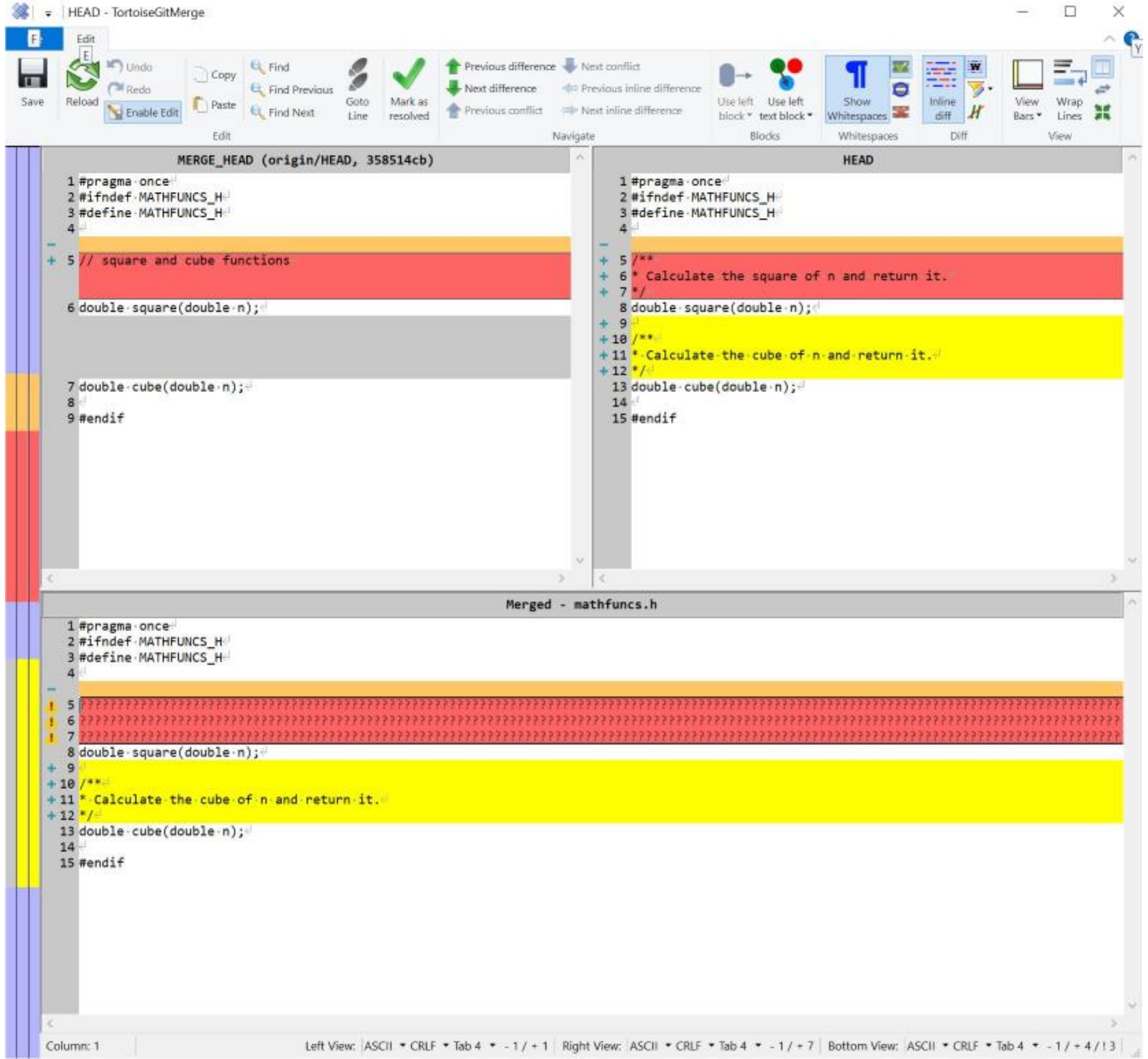
SimpleVSUnitTest/SimpleVSUnitTest/mathfuncs.h: Working Tree

```
  #pragma once
  #ifndef MATHFUNCS_H
  #define MATHFUNCS_H

+ /**
+ * Calculate the square of n and return it.
+ */
  double square(double n);
+
+ /**
+ * Calculate the cube of n and return it.
+ */
  double cube(double n);

  #endif
```

Conflicts – Two developers Change the Same Line



MERGE_HEAD (origin/HEAD, 358514cb)

```
#pragma once
#ifndef MATHFUNCS_H
#define MATHFUNCS_H

// square and cube functions
double square(double n);
double cube(double n);

#endif
```

HEAD

```
#pragma once
#ifndef MATHFUNCS_H
#define MATHFUNCS_H

/**
* Calculate the square of n and return it.
*/
double square(double n);

/**
* Calculate the cube of n and return it.
*/
double cube(double n);

#endif
```

Merged - mathfuncs.h

```
#pragma once
#ifndef MATHFUNCS_H
#define MATHFUNCS_H

??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
??????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????????
double square(double n);

/**
* Calculate the cube of n and return it.
*/
double cube(double n);

#endif
```

Column: 1 Left View: ASCII CRLF Tab 4 - 1 / + 1 Right View: ASCII CRLF Tab 4 - 1 / + 7 Bottom View: ASCII CRLF Tab 4 - 1 / + 4 / ! 3