

extracredit

0.1.0

Generated by Doxygen 1.8.17

1 doxygen-cmake-github	1
1.1 How to use	1
1.1.1 VS Code VM Instructions	1
1.1.2 General Usage	3
1.2 References	3
2 File Index	5
2.1 File List	5
3 File Documentation	7
3.1 /home/benson/CPTR227/ExtraCredit/README.md File Reference	7
3.2 /home/benson/CPTR227/ExtraCredit/src/main.cpp File Reference	7
3.2.1 Detailed Description	8
3.2.2 Function Documentation	8
3.2.2.1 main()	8
3.2.2.2 PrintBen()	9
Index	11

Chapter 1

doxygen-cmake-github

Demonstrates Doxygen html generation and publishing on GitHub Pages. The Doxygen files for this project can be seen [here](#).

1.1 How to use

1. Point your browser to this repository (<https://github.com/semcneil/doxygen-cmake-github>)
2. Press the "Use this template" button
3. Give your repository a new name
4. Write a short (one sentence) description of what your project will do
5. Click the Create repository from template button

1.1.1 VS Code VM Instructions

1. VS Code needs the following extension added:
 - (a) C/C++ from Microsoft
 - (b) CMake Tools also from Microsoft
2. Connect to Host in New Window
3. Open a terminal (`ctrl+``)
4. Initialize git if you haven't already using the same email you used on your GitHub account:
 - (a) `git config --global user.email "you@example.com"`
 - (b) `git config --global user.name "Your Name"`
5. Navigate to the parent directory for your project
6. Clone your repository using the URL from the GitHub Code button on your repository and on VS Code either clone repository on the Welcome screen or open the Command Palette (`ctrl+shift+P`), type `git clone` and select `Git: Clone`
 - (a) Select the parent directory for your project

- (b) Open the cloned repository either as prompted or by adding the newly created folder to your workspace by the Welcome tab's Open folder link or File -> Add Folder to Workspace
 - (c) If you use the command line `git clone` the authentications for pushing to your online repository are not set up
7. If you wait a bit it should ask you which kit you want to use (at the time of this writing I typically use GCC 9.3.0)
8. Allow Intellisense if prompted
9. Edit [README.md](#) to reflect your new project
10. Edit the `project` line in the `CMakeLists.txt` file to have your project's name and version
11. Edit the `add_executable` line in the `CMakeLists.txt` file to change the name of the executable file to something relevant
12. Change the `@brief`, `@details`, `@author`, and `@date` in [src/main.cpp](#)
13. To create the PDF on a standard Ubuntu install, the following need to be added: `sudo apt install graphviz texlive-latex-base texlive-latex-recommended texlive-latex-extra`
14. Doxygen also needs installing: `sudo apt install doxygen`
15. In the terminal, change to the `build` directory (should have been automatically generated)
16. Run the following:
 - (a) `make`
 - (b) `make docs`
 - (c) `make pdf`
17. Add the newly named PDF to git staging (`git status -> git add docs/yourprojectname.pdf`)
18. Commit all the changes: `git commit -a -m "Initial commit"`
19. Push the changes to GitHub: `git push origin main`
20. Back at your repository on GitHub, refresh the page to show latest commit
21. In the Settings tab, scroll down to GitHub Pages
22. Select "Branch: main" as source and `/docs` as the folder and then press Save
23. Scroll back down to GitHub Pages and click the link to the published site
24. You now have a C++ repository with doxygen output hosted on GitHub Pages
 - (a) The link usually doesn't work for a while (minutes to hours). This can be worked around by adding `index.html` to the end of the URL. A second commit will also fix it once the commit propagates over to GitHub Pages.
 - (b) You can see the PDF file generated by Doxygen by adding the name of the PDF to the end of the URL. It will be of the form `projectname.pdf` and can be seen in the `docs` folder.
 - (c) It can take a few minutes for a new `git push origin main` to propagate over to GitHub Pages
25. Edit [README.md](#) to reflect your project usage and point to the Doxygen output for your project
26. Stage the commit (`git add README.md`)
27. Commit (`git commit -a -m "Describe your changes here"`)
28. Push your changes to GitHub as before (`git push origin main`)

1.1.2 General Usage

During normal development, you will change [main.cpp](#), maybe add more files in the src directory, make them, and run them. To update the documentation on the web do the following at a terminal prompt in your project's build directory:

1. `make`
2. `make docs`
3. `make pdf` Then in your project's root directory do the following:
4. Check the git status: `git status`
5. `git commit -a -m "Describe your changes since last commit"`
 - (a) The `-a` flag is used to commit all the updated documentation files
 - (b) VS Code also has git built into it, but the use of branches isn't as easy a workflow as the commandline offers for me (personal opinion).
6. Note that in order for numbered (ordered) lists to work across markdown and Doxygen HTML and PDF outputs they are explicitly numbered vs markdown all being 1. or Doxygen's `-#`.

1.2 References

1. <https://www.doxygen.nl/manual/docblocks.html>
2. <https://stackoverflow.com/questions/44212101/cmake-how-to-have-add-custom-command-r>
3. Very useful overview: <https://caiorss.github.io/C-Cpp-Notes/Doxygen-documentation.%E2%9C%93.html>
4. <https://devblogs.microsoft.com/cppblog/clear-functional-c-documentation-with-sphinx>
5. <https://vicrucann.github.io/tutorials/quick-cmake-doxygen/>
6. <https://medium.com/practical-coding/c-documentation-with-doxygen-cmake-sphinx-breathe>
7. <https://stackoverflow.com/questions/18590445/cmake-custom-command-to-copy-and-rename>

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

<code>/home/benson/CPTR227/ExtraCredit/src/main.cpp</code>	
This is a test of CMake, doxygen, and GitHub	7

Chapter 3

File Documentation

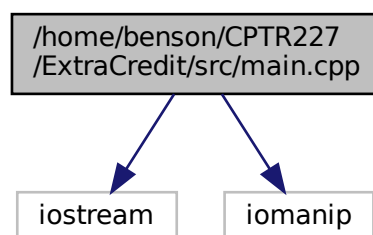
3.1 /home/benson/CPTR227/ExtraCredit/README.md File Reference

3.2 /home/benson/CPTR227/ExtraCredit/src/main.cpp File Reference

This is a test of CMake, doxygen, and GitHub.

```
#include <iostream>
#include <iomanip>
```

Include dependency graph for main.cpp:



Functions

- void `PrintBen` (int ben[][10], int n)
- int `main` ()

3.2.1 Detailed Description

This is a test of CMake, doxygen, and GitHub.

This is the long brief at the top of [main.cpp](#).

Author

Seth McNeill

Date

1/28/2021

3.2.2 Function Documentation

3.2.2.1 main()

```
int main ( )
```

Definition at line 30 of file main.cpp.

```
30     {
31         int i,j,v;
32         cout<<"Enter the number of vertexes: ";
33         cin>>v;
34         int ben[10][10];
35         cout<<"\n";
36
37         //Take input of the adjacency of each pair of vertexes
38         for(i=0;i<v;i++){
39             for(j=i;j<v;j++){
40                 if(i!=j){
41                     cout<<"Enter 1 if vertex "<i+1<" is adjacent to "<j+1<" , otherwise 0: ";
42                     cin>>ben[i][j];
43                     ben[j][i]=ben[i][j];
44                 }
45             }
46             else
47                 ben[i][j]=0;
48         }
49     }
50
51     }
52     }
53     PrintBen(ben,v);
54
55     return 0;
56
57
58 }
```

3.2.2.2 PrintBen()

```
void PrintBen (
    int ben[][10],
    int n )
```

Definition at line 15 of file main.cpp.

```
15 {
16     int i, j;
17     cout<<"\n\n"<<setw(4)<<" ";
18     for(i=0; i<n; i++)
19         cout<<setw(3)<<" ("<<i+1<<" ";
20     cout<<"\n\n";
21     //Print 1 if the corresponding vertexes are connected otherwise 0.
22     for(i=0; i<n; i++) {
23         cout<<setw(3)<<" ("<<i+1<<" ";
24         for(j=0; j<n; j++) {
25             cout<<setw(4)<<ben[i][j];
26         }
27         cout<<"\n\n";
28     }
29 }
```


Index

[/home/benson/CPTR227/ExtraCredit/README.md](#), [7](#)
[/home/benson/CPTR227/ExtraCredit/src/main.cpp](#), [7](#)

main
 [main.cpp](#), [8](#)
main.cpp
 [main](#), [8](#)
 [PrintBen](#), [8](#)

PrintBen
 [main.cpp](#), [8](#)