Options-Algo

0.4.0

Generated by Doxygen 1.8.17

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.↩ 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-breat
7. https://stackoverflow.com/questions/18590445/cmake-custom-command-to-copy-and-renam

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Class Documentation

4.1 options Class Reference

Public Member Functions

- options ()
- map< string, int > checker ()
- int total ()
- int average ()
- void stockprint ()
- void user (string ticker)

Public Attributes

- vector< string > check
- map< string, int > Porfolio
- map< string, int > values
- map< string, int > stock

4.1.1 Detailed Description

Definition at line 17 of file main.cpp.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 options()

```
options::options ( )  [inline]
```

Definition at line 25 of file main.cpp.

```
25             {
26
27         // Porfolio Data
28
29         Porfolio["APPL"] = 999997;
30         Porfolio["MSFT"] = 5230000;
31         Porfolio["BKB.A"] = 20000;
32         Porfolio["AMZN"] = 53000;
33         Porfolio["NVDA"] = 335000;
34
35
36
37         // Values Data
38
39         values["APPL"] = 300000;
40         values["TSLA"] = 500000;
41         values["MSFT"] =  458900;
42         values["AMD"] = 234000;
43         values["COIN"] = 543200;
44         values["GOOGL"] = 980889;
45         values["FB"] = 234312;
46         values["NVDA"] = 2884003;
47         values["SPY"] = 2000;
48         values["P"] = 5432090;
49         values["BKB.A"] = 9809;
50         values["BKB.B"] = 234312;
51         values["G"] = 213131;
52         values["AMZN"] = 2300;
53         values["ALP"] = 54300;
54         values["N"] = 9808;
55         values["BK"] = 23492;
56         values["B"] = 21312;
57
58     }
```

4.1.3 Member Function Documentation

4.1.3.1 average()

```
int options::average ( )  [inline]
```

Definition at line 92 of file main.cpp.

```
92             {
93     int average;
94     int amount = total();
95     average = amount/stock.size();
96
97     return average;
98 }
```

4.1.3.2 checker()

```
map<string, int> options::checker ( )  [inline]
```

Definition at line 60 of file main.cpp.

```
60                          {
61     int val;
62     string key;
63     map<string, int>::iterator j;
64
65     for(j=Porfolio.begin(); j != Porfolio.end();j++){
66         key = j->first;
67         if(values.find(key) != values.end()){
68                 //Add it to the stock map both the key and value
69                 val = values[key];
70                 stock[key] = val;
71         }
72     }
73     return stock;
74 }
```

4.1.3.3 stockprint()

```
void options::stockprint ( )  [inline]
```

Definition at line 100 of file main.cpp.

```
100                  {
101     map<string, int>::iterator k;
102     map<string, int>::iterator i;
103     int count = 0;
104     int avg = average();
105
106     cout « "List of stocks in porfolio and amount of contracts held" « endl;
107
108     for(i = Porfolio.begin(); i != Porfolio.end();i++){
109         cout « ++count « "." «" " « i->first « " " « "=" « " " « Porfolio[i->first] « "." « endl;
110     }
111
112     cout « "List of stocks to expire and amount of contracts" « endl;
113
114     count = 0;
115     for(k = stock.begin(); k != stock.end();k++){
116         cout « ++count « "." «" " « k->first « " " « "=" « " " « stock[k->first] « "." « endl;
117     }
118
119     cout « "The following stocks in your porfolio have higher amount of contracts than the average
    amount of contracts to expire" « endl;
120
121     count = 0;
122     for(k = Porfolio.begin(); k != Porfolio.end();k++){
123         if(avg < Porfolio[k->first]){
124             cout «  ++count « ". " « k->first «endl;
125             check.push_back(k->first);
126
127         }
128     }
129
130     cout « "Enter the ticker of which stock you want to sell: " « endl;
131
132
133 }
```

4.1.3.4 total()

```
int options::total ( )  [inline]
```

Definition at line 76 of file main.cpp.

```
76           {
77     map<string, int> data; //map assigned for the returned value of checker function
78     map<string, int>::iterator i;
79     int count = 1;
80     data = checker();
81     int total = 0;
82 
83     //calculating total contracts to be expired.
84     for(i = stock.begin(); i != stock.end();i++){
85         total += data[i->first];
86     }
87 
88     return total;
89 
90 }
```

4.1.3.5 user()

```
void options::user (
             string ticker )  [inline]
```

Definition at line 137 of file main.cpp.

```
137                         {
138 
139     map<string, int>::iterator k;
140     int sell;
141     int count = 1;
142     int avg = average();
143     int left;
144 
145     //checking for input errors
146     for(int i = 0; i<check.size(); i++){
147         if(!Porfolio[ticker]){
148             cout « "Invalid value" « endl;
149             break;
150         }else{
151             cout « "calculating the average differences... " « endl;
152 
153             //calculating differences between the average contracts to be expired and currently owned
      contracts of each stock in porfolio.
154             for(k = Porfolio.begin(); k != Porfolio.end();k++){
155                 if(avg < Porfolio[ticker]){
156                     sell = Porfolio[ticker] - avg;
157                     left = Porfolio[ticker] - sell;
158                     cout « count++ « ". " « sell « " amount of " « ticker « " has been sold." « endl;
159                     cout « "You have " « left « " contracts left of stock " « ticker « endl;
160                     break;
161                 }
162             }
163         break;
164         }
165     }
166 }
```

4.1.4 Member Data Documentation

4.1.4.1 check

```
vector<string> options::check
```

Definition at line 19 of file main.cpp.

4.1.4.2 Porfolio

```
map<string, int> options::Porfolio
```

Definition at line 20 of file main.cpp.

4.1.4.3 stock

```
map<string, int> options::stock
```

Definition at line 22 of file main.cpp.

4.1.4.4 values

```
map<string, int> options::values
```

Definition at line 21 of file main.cpp.

The documentation for this class was generated from the following file:

- /home/addis/Options-Algo/src/main.cpp

Chapter 5

File Documentation

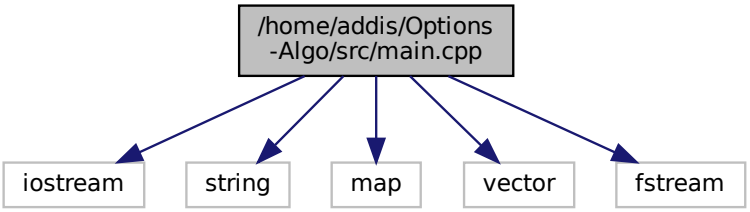
5.1 /home/addis/Options-Algo/README.md File Reference

5.2 /home/addis/Options-Algo/src/main.cpp File Reference

This is a programs that checks amount of options to be expired and optimizes the poroflio.

```
#include <iostream>
#include <string>
#include <map>
#include <vector>
#include <fstream>
```

Include dependency graph for main.cpp:



Classes

- class options

Functions

- int main (int, char **)

5.2.1 Detailed Description

This is a programs that checks amount of options to be expired and optimizes the poroflio.

Author

Addis Bogale and Bona Tufa

Date

04/21/2021

5.2.2 Function Documentation

5.2.2.1 main()

```
int main (
              int ,
              char **  )
```

Definition at line 172 of file main.cpp.

```
172                       {
173     string input;
174
175     options test;
176     test.checker();
177     test.stockprint();
178     cin » input;
179     test.user(input);
180
181 }
```

Index