Options-Algo

0.4.0

Chapter 1

Options-algo

This program demonestrates how to optimize option contracts porfolio according to the average of amount expiring option contracts expiring from a data set given by the developer.

1.1 Why

The finance industry is growing in tremendous speed by using technology. Project like this play fundamental roles in quantitative finance firms.

1.2 How to use

1. The program source code can be found at (`https://github.com/addisgithub/Options-↩ Algo.git`) or ()

2. After cloning the templete, you can run the pprogram from the main file in src.

1.2.1 Instructions & Examples

Although the data is given by the developer you can edit the file as needed. The two maps used to show the data are portfolio & values both can be altered and are found in the constuctor of the class.

1.3 Data structures & Agorithms

1. Maps are used frequently in this program because of their key value paring structure. They are used to match between differnt data sets using their keys.

2. Vectors are used in this program for storage of strings(tickers).

3. The algorithm calculates the average amount of expiring contracts in accordance to our porfolio and optimizes our porfolio to maintain eual or below market average.

4. Since this data was made by the developer it was easier to traverse through using loops but if it was a larger data set pulled by an API of actual stock market option chain data set then we can use a graph structure and traverse through it using bfs or dfs.

1.4 References

1. https://www.geeksforgeeks.org/

2. https://www.cplusplus.com/reference/algorithm/find/

3. https://stackoverflow.com/

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

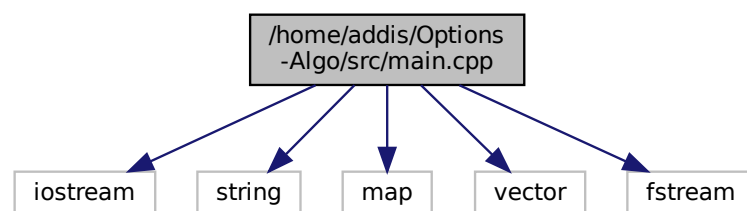# Chapter 5

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 by average amount of contracts to expire.

```
#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 by average amount of contracts to expire.

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