

## MASTERMIND STL PROJECT

### Introduction:

I decided to code mastermind as my game choice because I coded it last semester in CSC7 and didn't use any STL components. Therefore, I knew it would be a fun challenge to rework it by adding STL functionality. I also added another simple feature to the game that my CSC7 version did not have. I spent around 8-10 hours reworking the code and thinking through how I can add the necessary STL components. I wrote the program procedurally versus taking an OOP approach and using classes.

Lines of Code: ~500

Classes: None

Github: <https://github.com/JCarlito/17C-Mastermind-STL-Project>

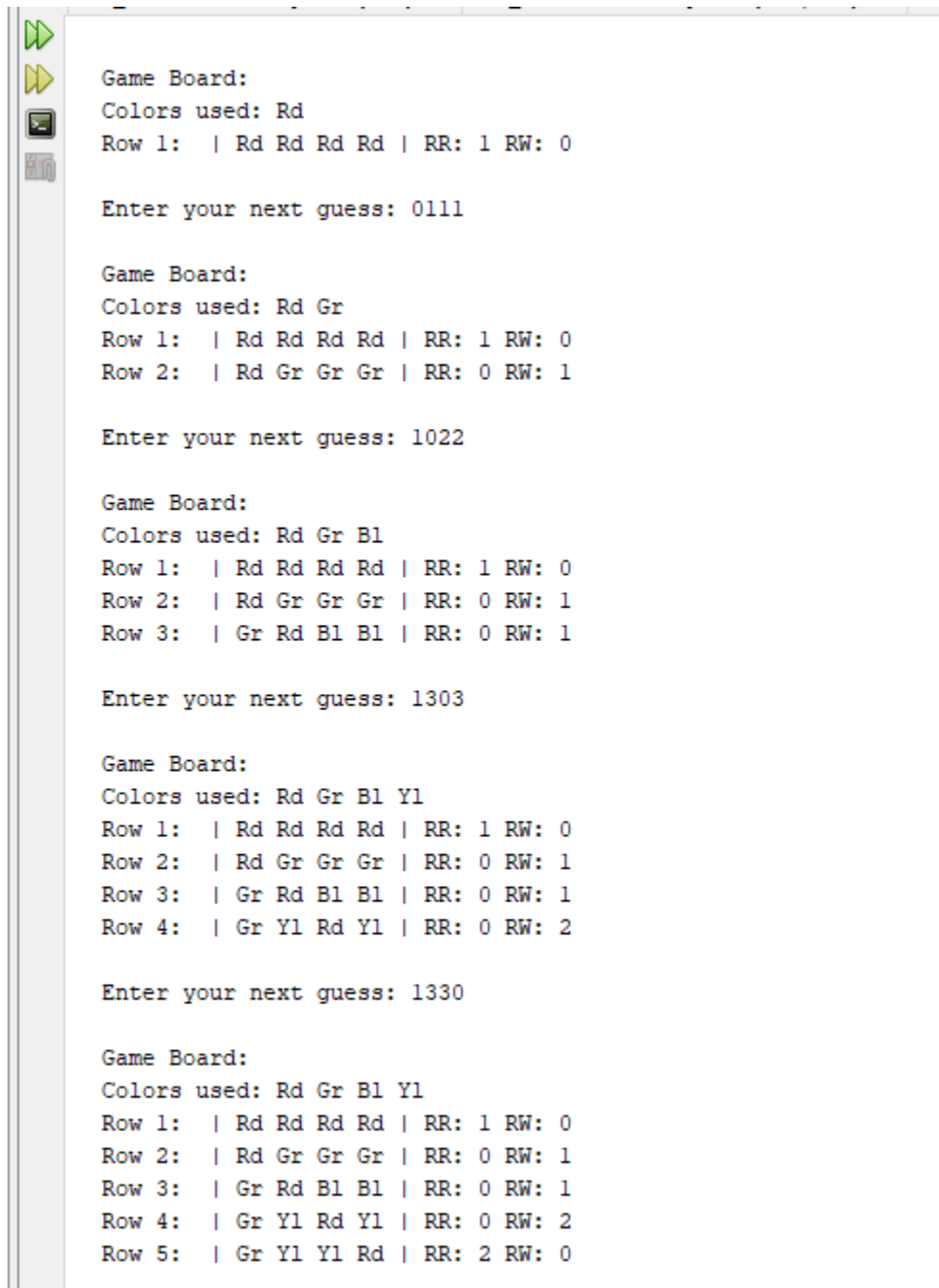
### Development Approach:

I took a procedural approach in my project. I used git and put my versions up on github. However, I did create completely separate versions when adding different STL components to help me work through the design instead of trying to write everything in one go. In version 0, I used a list and iterators to shuffle around and create the code that needs to be solved. In version 1, maps were used to calculate the right color but wrong place (RW) hints. It stores the color and its occurrences in both a code map and a guess map. It then compares to similar keys and takes the min value out of the keys and accumulates that in (RW). In version 2, I used queues/stacks to display the gameboard instead of a 2 dimensional array. The queues are used to display in normal mode. The stacks are used to display in practice mode where the user can remove their previous guess to continue playing and not run out of turns to get the hang of the game. In version 3, I fixed issues with hints not properly being displayed in practice mode. The mastermind game rules are simple. The player has to break a code which is displayed in the form of ordered colors. There are 8 colors in total that can be used to create the code. The user has the option of guessing a 4, 6, or 8 color code, if they want duplicate colors in their code or not, and how many guesses they'd like to give themselves from 8, 10, or 12. As the user guesses the game gives back hints. The hints are the number of right colors in the right place and the number of right colors in the wrong place. The game ends when the user correctly guesses the color code or runs out of turns.

## Description:

The code is composed primarily of functions. In main() there are two branches that signify game modes. The first branch is a normal mode game loop and the second branch is a practice mode game loop where the user can delete previous guesses. No classes were used in this project.

Normal Mode:



```
Game Board:
Colors used: Rd
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0

Enter your next guess: 0111

Game Board:
Colors used: Rd Gr
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Rd Gr Gr Gr | RR: 0 RW: 1

Enter your next guess: 1022

Game Board:
Colors used: Rd Gr Bl
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Rd Gr Gr Gr | RR: 0 RW: 1
Row 3:  | Gr Rd Bl Bl | RR: 0 RW: 1

Enter your next guess: 1303

Game Board:
Colors used: Rd Gr Bl Yl
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Rd Gr Gr Gr | RR: 0 RW: 1
Row 3:  | Gr Rd Bl Bl | RR: 0 RW: 1
Row 4:  | Gr Yl Rd Yl | RR: 0 RW: 2

Enter your next guess: 1330

Game Board:
Colors used: Rd Gr Bl Yl
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Rd Gr Gr Gr | RR: 0 RW: 1
Row 3:  | Gr Rd Bl Bl | RR: 0 RW: 1
Row 4:  | Gr Yl Rd Yl | RR: 0 RW: 2
Row 5:  | Gr Yl Yl Rd | RR: 2 RW: 0
```

Practice mode (-1 signifies a turn being deleted)



```
Game Board:
Colors used: Rd
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
```

Enter your next guess: 1111

```
Game Board:
Colors used: Rd Gr
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Gr Gr Gr Gr | RR: 0 RW: 0
```

Enter your next guess: 2222

```
Game Board:
Colors used: Rd Gr Bl
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Gr Gr Gr Gr | RR: 0 RW: 0
Row 3:  | Bl Bl Bl Bl | RR: 1 RW: 0
```

Enter your next guess: -1

```
Game Board:
Colors used: Rd Gr Bl
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Gr Gr Gr Gr | RR: 0 RW: 0
```

Enter your next guess: -1

```
Game Board:
Colors used: Rd Gr Bl
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
```

Enter your next guess: 0345

```
Game Board:
Colors used: Rd Gr Bl Yl Br Or
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Rd Yl Br Or | RR: 1 RW: 1
```

Enter your next guess: 0637

```
Game Board:
Colors used: Rd Gr Bl Yl Br Or Bk Wh
Row 1:  | Rd Rd Rd Rd | RR: 1 RW: 0
Row 2:  | Rd Yl Br Or | RR: 1 RW: 1
Row 3:  | Rd Bk Yl Wh | RR: 1 RW: 1
```

## Check Off Sheet:

### LINES REFERENCE V3

#### Containers:

1. Sequences
  - a. List
    - i. Used in the setCode() function to hold the possible color numeric values and was shuffled in order to produce a non duplicate code
    - ii. Lines 323-347
2. Associative Containers
  - a. Set
    - i. Used to store how many colors a player has already used in their code and then output at the top of each game board.
    - ii. Declared on line 53 and updated throughout both game mode loops
  - b. Map
    - i. Used in the hints() function in order to increment the RW hint
    - ii. Lines 351-378
3. Container adaptors
  - a. Stack
    - i. Used to store guesses and hints in practice mode
    - ii. Seen throughout all of the practice mode game loop branch and displayGameboard() function
  - b. Queue
    - i. Used to store guesses and hints in normal mode
    - ii. Seen throughout all of the normal mode game loop branch and displayGameboard() function

#### Iterators:

1. List
  - a. Bidirectional
2. Maps
  - a. Bidirectional Iterator
3. Sets
  - a. Bidirectional Iterator
4. Stacks (Depends on the container that was adapted)
  - a. Bidirectional Iterator
  - b. Random Access Iterator
5. Queues (Depends on the container that was adapted)
  - a. Bidirectional Iterator
  - b. Random Access Iterator

### Algorithms:

1. Non-mutating algorithms
  - a. Find()
2. Mutating algorithms
  - a. Swap()
  - b. Reverse()
3. Organization
  - a. Min()

**Pseudocode:**

*Initialize variables*

*Display game intro*

*Get user inputs: nColors, gameLen, dup, modeFlag*

*IF game mode is Normal:*

*Generate random code*

*Print generated code*

*Get user's guess*

*Validate guess length*

*Add colors to usedColors*

*Add guess to guesses queue*

*WHILE guesses less than gameLen AND user hasn't guessed code:*

*Evaluate user's guess and provide hints*

*Store hints*

*Display game board*

*Get user's next guess*

*Validate guess length*

*Add colors to usedColors*

*Add guess to guesses queue*

*Evaluate user's last guess and provide hints*

*Store hints*

*Display game board*

*Display outro message*

*IF game mode is Practice:*

*Print deletion message*

*Generate random code*

*Print generated code*

*Get user's guess*

*Validate guess length and check for deletion attempt*

*Add colors to usedColors*

*Add guess to practiceGuesses stack*

*WHILE practiceGuesses less than gameLen AND user hasn't guessed code:*

*Evaluate user's guess and provide hints (skip if deleting)*

*Store hints*

*Display game board*

*Get user's next guess*

*Validate guess length and check for deletion attempt*

*IF deleting, pop practiceGuesses stack and continue*

*Add colors to usedColors*

*Add guess to practiceGuesses stack*

*Evaluate user's last guess and provide hints (skip if deleting)*

*Store hints*

*Display game board*

*Display outro message*

*Exit program*

## Flowchart:

