CS 162 Project 1 Design Plan

Brittany Dunn

April 15, 2018

❖ Menu function

```
Int main()
{
    Char choice;
    displayStartMenu();
    choice = getChoice();
    //if choice 1 simulaton or If 2 quit
    displayEndMenu();
    choice = getChoice();
    //if choice 1 simulaton or If 2 quit

        //Menu function(hpp/cpp)
        {
                Void displayStartMenu()
                {
                        Cout << 1. Start
                                2. Quit
                                Please enter the number of your choice
                }

                Void displayEndMenu()
                {
                        Cout << 1. Play again
                                2. Quit
                                Please enter the number of your choice
                }
                Char get Choice()
                {
                        char choice = cin.get
                        cin.ignore();
                        while(choice < ‘1’ or choice > ’2’)
                        {
                        Cout << Choice must be 1 or 2
                                <<Please  enter number of your chocie
                        Choice = cin.get();
                        Cin.ignore();
                        }
                        Return choice;
        }
}

    if (choice == ‘1’)
```

❖ Start

```
    //startGame Function (hpp/cpp)
    {
```

- ▪ Get User Input

Int row
Int column
Int steps
Int startRow
Int startColumn

- Number of rows

Please enter The number of rows for the board.
Cin << row

  Input Validation function

  - Validate
    - > 0
    - Integer
- Number of column

Please enter The number of columns for the board.
Cin << column

  Input Validation function

  - Validate
    - > 0
    - integer
- Number of steps

Please enter The number of steps during simulation.
Cin << steps

  Input Validation function

  - Validate
    - > 0
    - integer
- Start row

Please enter The starting row of the ant.
Cin << startrow
startRow -= 1

  Input Validation function

  - Validate
    - Within board
    - integer
- Start column

Please enter The starting column of the ant.
Cin << startcolumn
StartColumn -= 1

  Input Validation function

  - Validate
    - Within board

- Integer

```
//Input Validation Function (hpp/cpp)
    Int getInt(input string)
    {
        String = inputValid(input)
        Int input = stoi(input,nullptr,0)
        Return input
    }
    String inputValid(input string)
    {
        For(i=0; i<input.length;++)
            If(!isdigit(input[i])
            {
                Invalid please enter new
                Cin >> input
            }
    }
```

- Allocate dynamtic memory for 2D array

  Use for loop to create new
- Free Allocated dynamtic memory for 2D array

  Use for loop to delete
- Create ant object

//Ant Class(hpp/cpp)

- Ant Class
  - Constructor

```
Ant(int,int,int,int,int,ptr**)
    max Row = row
    maxColumn = columns;
    rows = start row;
    columns = start columns;
    steps = steps;
    board = ptr**;
    for(int row = 0; row < maxRow; ++row)
    {
        for(int col = 0; col < maxColumn; ++col)
        {
            board[row][col] = ' ';
        }
    }
```

```
        board[rows][columns] = '*';
        color = ' ';
        antFacing = NORTH;
        currentStep = 0;
```

- Data Members (private)
  - Steps

    Int steps
  - Current step

    Int currentStep
  - current Row

    Int row
  - current Column

    Int columns
  - Board 2D array

    Char** board
  - Color

    Char color
  - Direction (north,south,east,west)

    Enum direction
  - Max rows

    Int maxRows
  - max columns

    int maxcolumns
- Functions(private)
  - change direction

```
Direction changeDirectionRight()
{
If statements to rotate 90 right
Return rotated direction
}
Direction changeDirectionLeft()
{
If statements to rotate 90 left
Return rotated direction
}
```

  - make move (Langton’s Ant rule) – needs edge cases

```
Void makeMove()
{
If(color == “ “)
        {
        Board[row][column] = “#”
        Direction = changeDirectionRight();
        switch(direction)
        { case for each N,S,E,W
                ~~If(row or column +/-1 <= maxrow/maxcol)~~
                        Adjust row or col +/- 1 based on change
```

```
                    ~~Else(row/col +/- 1 > maxrow/maxcol)~~
                            ~~changeDirectionRight();~~
             Break
            }


If(color == “#”)
        {
        Board[row][column] = “ “
        Direction = changeDirectionLeft();
        switch(direction)
        { case for each N,S,E,W
                ~~If(row or column +/-1 <= maxrow/maxcol)~~
                        Adjust row or col +/- 1 based on change
                ~~Else(row/col +/- 1 > maxrow/maxcol)~~
                        ~~changeDirectionRight();~~
         Break
        }
```
If(row/column outside of board)

```
        {
        Move ant to board[0][0]
        Row=0
        Column=0
        }
Color = board[row][column]
Board[row][column] = “*”
```

- Functions (public)
    - Start simulation

```
Void startSimulation()
{
If(current step <= steps)
        {
        Board.makeMove();
        Board.printBoard();
        }
}
```

    - Print board

```
Void printBoard()
{
For loop to print board
Add _ | (edge lines)
}
```

if (choice == ‘2’)

- Quit

Location	Test Case	Input	Expected Output	Actual Output
Start Menu displayStartMenu(); getChoice();	1	1	Start Program	Start Program
Start Menu displayStartMenu(); getChoice();	2	2	Quit Program	Quit Program
Start Menu displayStartMenu(); getChoice();	Integer not 1 or 2	8	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Start Menu displayStartMenu(); getChoice();	Character	t	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Start Menu displayStartMenu(); getChoice();	String (only char)	abc	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Start Menu displayStartMenu(); getChoice();	String(char digit)	a7	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Start Menu displayStartMenu(); getChoice();	String (digit char)	5d	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Start Menu displayStartMenu(); getChoice();	String(char digit char)	s5s	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid

Location	Test Case	Input	Expected Output	Actual Output
End Menu displayEndMenu(); getChoice();	1	1	Start Program	Start Program
End Menu displayEndMenu(); getChoice();	2	2	Quit Program	Quit Program
End Menu displayEndMenu(); getChoice();	Integer not 1 or 2	8	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
End Menu displayEndMenu(); getChoice();	Character	q	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
End Menu displayEndMenu(); getChoice();	String (only char)	abc	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
End Menu displayEndMenu(); getChoice();	String(char digit)	j7	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
End Menu displayEndMenu(); getChoice();	String (digit char)	0g	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
End Menu displayEndMenu(); getChoice();	String(char digit char)	d3f	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid

Location	Test Case	Input	Expected Output	Actual Output
Get Number of Rows row = getInt(temp)	Integer greater than 0	5	move to next step	move to next step
Get Number of Rows row = getInt(temp)	0	0	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Rows row = getInt(temp)	integer less than 0	-10	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Rows row = getInt(temp)	Character	a	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Rows row = getInt(temp)	String (only char)	abc	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Rows row = getInt(temp)	String(char digit)	d4	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Rows row = getInt(temp)	String (digit char)	8d	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Rows row = getInt(temp)	String(char digit char)	y5x	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid

Location	Test Case	Input	Expected Output	Actual Output
Get Number of Columns column = getInt(temp)	Integer greater than 0	2	move to next step	move to next step
Get Number of Columns column = getInt(temp)	0	0	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Columns column = getInt(temp)	integer less than 0	-88	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Columns column = getInt(temp)	Character	h	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Columns column = getInt(temp)	String (only char)	dsf	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Columns column = getInt(temp)	String(char digit)	q7	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Columns column = getInt(temp)	String (digit char)	9r	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Columns column = getInt(temp)	String(char digit char)	w5j	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid

Location	Test Case	Input	Expected Output	Actual Output
Get Number of Steps steps = getInt(temp)	Integer greater than 0	9	move to next step	move to next step
Get Number of Steps steps = getInt(temp)	0	0	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Steps steps = getInt(temp)	integer less than 0	-66	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Steps steps = getInt(temp)	Character	b	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Steps steps = getInt(temp)	String (only char)	bdcs	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Steps steps = getInt(temp)	String(char digit)	g6	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Steps steps = getInt(temp)	String (digit char)	3k	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Number of Steps steps = getInt(temp)	String(char digit char)	l7p	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid

Location	Test Case	Input	Expected Output	Actual Output
Get Starting Row startingRow = getInt(temp)	Integer greater than 0	3	move to next step	move to next step
Get Starting Row startingRow = getInt(temp)	Integer equal to rows	3	move to next step	move to next step
Get Starting Row startingRow = getInt(temp)	0	0	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Row startingRow = getInt(temp)	integer less than 0	-2	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Row startingRow = getInt(temp)	integer greater than rows	4	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Row startingRow = getInt(temp)	Character	e	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Row startingRow = getInt(temp)	String (only char)	efg	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Row startingRow = getInt(temp)	String(char digit)	s2	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Row startingRow = getInt(temp)	String (digit char)	3x	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Row startingRow = getInt(temp)	String(char digit char)	n8m	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid

Location	Test Case	Input	Expected Output	Actual Output
Get Starting Column startingColumn = getInt(temp)	Integer greater than 0	4	move to next step	move to next step
Get Starting Column startingColumn = getInt(temp)	Integer equal to columns	4	move to next step	move to next step
Get Starting Column startingColumn = getInt(temp)	0	0	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Column startingColumn = getInt(temp)	integer less than 0	-7	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Column startingColumn = getInt(temp)	integer greater than columns	5	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Column startingColumn = getInt(temp)	Character	i	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Column startingColumn = getInt(temp)	String (only char)	tre	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Column startingColumn = getInt(temp)	String(char digit)	r3	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Column startingColumn = getInt(temp)	String (digit char)	8m	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid
Get Starting Column startingColumn = getInt(temp)	String(char digit char)	k6l	Prompt user to enter new choice until valid	Prompt user to enter new choice until valid

Location	Test Case	Input
Ant.cpp startSimulation();	simulation follows Langton's Ant Rules correctly	startRow = 3 startColumn = 3 steps = 5 rows = 6 columns = 5

Expected Output

```
#   *
```

current step	1
direction	E
color	W
row	2
column	3

```
#   #
    *
```

current step	2
direction	S
color	W
row	3
column	3

Actual Output

```
#   *
```

current step	1
direction	E
color	W
row	2
column	3

```
#   #
    *
```

current step	2
direction	S
color	W
row	3
column	3

```
    #   #
    *   #
```

current step	3
direction	W
color	W
row	3
column	2

```
    *   #
    #   #
```

current step	4
direction	N
color	#
row	2
column	2

```
*       #
    #   #
```

current step	5
direction	W
color	W
row	2
column	1

```
    #   #
    *   #
```

current step	3
direction	W
color	W
row	3
column	2

```
    *   #
    #   #
```

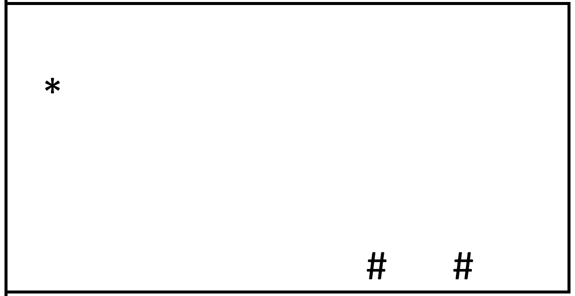
current step	4
direction	N
color	#
row	2
column	2

```
*       #
    #   #
```

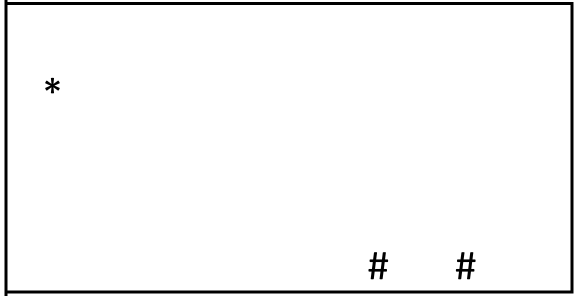
current step	5
direction	W
color	W
row	2
column	1

Location	Test Case	Input
Ant.cpp startSimulation();	simulation follows Langton's Ant Rules correctly and edge cases cause ant to return to 0,0 of board	startRow = 5 startColumn = 4 steps = 5 rows = 5 columns = 6

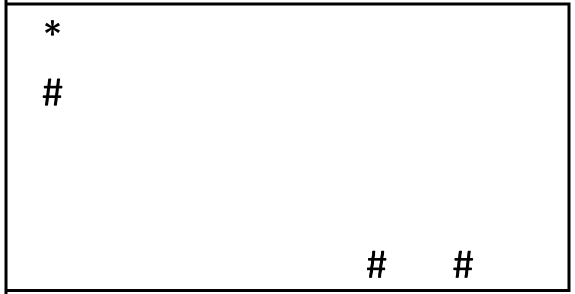
Expected Output			**Actual Output**		
	current step	1		*current step*	1
	direction	E		*direction*	E
	color	W		*color*	W
	row	4		*row*	4
# *	*column*	4	# *	*column*	4
start 54 43			start 54 43		
Rows 5 Columns 6 Steps 5			Rows 5 Columns 6 Steps 5		
*	*current step*	2	*	*current step*	2
	direction	S		*direction*	S
	color	W		*color*	W
	row	0		*row*	0
# #	*column*	0	# #	*column*	0
*	*current step*	3	*	*current step*	3
	direction	W		*direction*	W
	color	#		*color*	#
	row	0		*row*	0
# #	*column*	0	# #	*column*	0



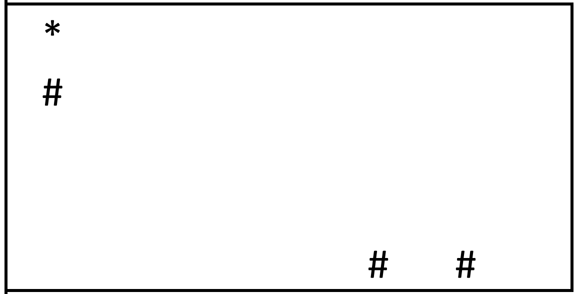
current step	4
direction	S
color	W
row	1
column	0



current step	4
direction	S
color	W
row	1
column	0



current step	5
direction	W
color	W
row	0
column	0



current step	5
direction	W
color	W
row	0
column	0

CS 162 Project 1 Reflection

Brittany Dunn

April 15, 2018

As I was writing the code for project 1 I ran into many problems and had to change my design plan accordingly. The main areas I ran into issues where with the input validation, the menu and how to handle the edge cases for the board. There were many small issues along the way which were easy fixes when I found where the error was occurring.

I spent a majority of time for this project on input validation. I have not had to do input validation before and I have no previous coding experience other than what I learned in CS 162. I was not sure even where to start with input validation. Originally in writing my design plan I just added a line if the input needed to be validated. I needed to do some research before I could write pseudocode for that section. So, I began researching input validation. First, I tried using the resources provided on canvas. I found these only slightly helpful. They involved coding in C instead of C++ which made them a little confusing. The example was also for validating floats which I could not figure out how to modify it to work for integers or characters. When that was not enough information I went to piazza to see if anyone else was struggling with this as well. There I found a post which mentioned checking strings for the input you want and then converting the strings when you know they contain only what you need to the data type you need. This post mentioned stoi which I had not heard of before. This made sense to me so my next step was looking up stoi. After some research I figured out how to use stoi. At this point I felt I had enough information to write pseudocode so I went back to my design plan and added it in. Then I began coding.

I ran into issues when trying to code the input validation. I had made two functions: getInt and inputValid. My idea was getInt would be passed the user's input as a string and then inputValid would make sure it was the correct data and then getInt would convert the input to the correct data type. I learned quickly this did not work but had a hard time figuring out why. I realized the flow of data back and forth between these functions and the coding calling these functions was the issue. I decided to separate the functions out so they were not calling each other as much and instead used the main function to call the functions. This was so the functions all had only one purpose and the data was not being passed so much the variables needed were out of scope or were the wrong ones being accessed. This resolved the issues I was having.

I had many issues with the menu as well. The first one involved input validation. I originally was trying to use a function called getChoice to confirm the choice was valid. I was trying to validate the user input was a char. I had used the book to help me figure out this input validation. This function would work for the startMenu but I could not get it to work a second time for the endMenu. This was one of the first things I wrote for this project, so I decided to leave this problem for after I coded everything else. By the time I went back to this I had a better understanding of input validation because I had done it in other areas of the program. I realized I had overcomplicated the input validation again as I had in other areas of this project. So, I changed the functions again. I realized I was relying too much on the main function for the menu. I was not making my menu reusable which was my other issue with the menu.

Once I realized I needed to move more of the menu out of main to fix the validation issue and make the menu more reusable, I reworked my getChoice function. I changed it from being only input validation to be a switch statement which can be changed based on the menu. In this switch statement I was able to create a default of invalid input. This simplified the input validation because now if neither of the cases was matched I could easily prompt the user for a new input. Thus, resolving my input validation issue and menu reusability issue.

The other main problem I had was with the edge cases of the ant. Originally, I was going to have the ant rotate once more when it was about to go off the edge. The issue I had was with how I was keeping track of color and location of the ant. If I had the ant rotate once more then it would end up just rotating back and forth in the same spot for the rest of the spots. This is because no matter what the step I always switched the color. I tried fixing this unsuccessfully and decided I would instead change how end cases were handled. So, I decided if the ant went off the board it would always be returned to board[0][0]. This I was able to code without major issues.