Adam Wright

CS-162
Langton's Ant: Design and Reflection

Program Design

The problem is to define the current position of the ant within the 2D array and to also store the ant's direction that it is facing. Then as the ant progresses through the pattern of moves that depend on the color of the square that it is standing on, the current position and direction information will need to be updated, along with the color of the current square. My though is that the 2D array will hold the space characters and pound symbols and then the asterisk for the ant will just be displayed as a cout and not stored on the board. The ant x and y will be stored as their own variables. The direction can be stored like an enum representing the 4 cardinal directions. When the ant hits and edge, which would either be 0 or the size of the dimension, then I like the choice of turning the ant 180 degrees in direction and letting the move function continue as before. I know that the y axis prints double spaces and the printing of a board is distored if an extra character isn't added to each inner for loop in the x axis. Since the x axis only has a bit more than 80 characters before it wraps, I plan to make the largest board 40 spaces wide so that an extra character can be added to make the final width 80 by 80 for a 40 by maximum board size. I think that 2 by 2 should be the smallest meaningful board. There should be a second menu function in the menu class that will be called after the simulation runs and the second function will return a value to a do while loop in main to allow the program to either quit or to run the program again.

Pseudocode:

```
main will create an instance of menu menu cout a welcome to Langton's ant message

cout 3 option menu for automatic ant position / manual ant / quit

    while loops to make sure that input is correct

    entering 3 will exit

    entering 1 sets a var for auto ant placement

    else 2 will set manually

    cin values for the number of rows and columns

    cin value for number of steps

    if manually setting ant - cin start row and column

        menu class passed into ant class to use getters for the set values

        ant class instance created in main

        ant will create a new 2d array based on the values from menu

        nested for loop to fill it with spaces

        remaining functions in ant called from a for loop up to num steps

        change color then turn left function or turn right function
        edge check function

        move space function

        double for loop to print board

    call final menu function and return value to main
```

Test Table:

Testing Input	Expected Output	Actual Output
run program main	calls menu and prints	calls menu and prints
main calls menu	menu prints enter 1 or 2 or 3	menu prints enter 1 or 2 or 3
User enters 3	Program quits	program quits
User enters 1 to choose auto ant	program continues to the next prompt	program continues to the next prompt
user enters a char user enters a num other than 1 or 2 or 3 user enters a string	propmted to enter only 1 or 2 or 3 prompted to enter either 1 or 2 or 3 prompted to enter either 1 or 2 or 3	prompted to enter either 1 or 2 or 3 prompted to enter either 1 or 2 or 3 prompted to enter either 1 or 2 or 3
User enters 2 to choose manual ant	program continues to the next prompt	program continues to the next prompt
user enters a char user enters a num other than 1 or 2 or 3 user enters a string	propmted to enter only 1 or 2 or 3 prompted to enter either 1 or 2 or 3 prompted to enter either 1 or 2 or 3	prompted to enter either 1 or 2 or 3 prompted to enter either 1 or 2 or 3 prompted to enter either 1 or 2 or 3
following Menu prompts	user asked for num rows	user asked for num rows
enters num between 2 - 40 user enters a char user enters a num outside 2 - 40 user enters a string	moves to num columns propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40	moves to num columns propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40
following Menu prompts	user asked for num columns	user asked for num columns
enters an num between 2 - 40 user enters a char user enters a num outside 2 - 40 user enters a string	moves to num steps propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40	moves to num steps propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40
following Menu prompts	user asked for num steps	user asked for num columns
enters an num between 2 - 20000 user enters a char user enters a num outside 2 - 20000 user enters a string	moves to start col if manual ant or to ant class if auto prompted to enter between 2 and 20000 prompted to enter between 2 and 20000 prompted to enter between 2 and 20000	moves to start col if manual ant or to ant class if auto prompted to enter between 2 and 20000 prompted to enter between 2 and 20000 prompted to enter between 2 and 20000
Menu prompt if manual ant	user asked for start row	user asked for start row
enters an num between 2 - 40 user enters a char user enters a num outside 2 - 40 user enters a string	moves to start column propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40	moves to start column propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40
Menu prompt if manual ant	user asked for start column	user asked for start column
enters an num between 2 - 40 user enters a char user enters a num outside 2 - 40 user enters a string	moves to ant class propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40	moves to ant class propmted to enter between 2 and 40 propmted to enter between 2 and 40 propmted to enter between 2 and 40
Ant class	compiles when called	compiles when called
User user menu values User user menu values	user menu values rendered dot between each grid space	user menu values rendered dot between each grid space
end menu to play agin or quit	choose 1	
User enters 1 User enters 2	return to main menu quit	return to main menu quit

Reflection:

I found that while I ran into many problems that I could not solve in the way that I wanted, in every case there was an uglier answer that came to my mind that would work instead. I planned to separate my wall checking, my right turning, left turning and move space functions into their own helper function, but I could not make the function calls work. I cut the code out of them and pasted it into the places of the function calls and then everything sprung to life, so I decided to just leave it as it was. It seemed ugly but after having tried to get the function calls to work I was happy to have code that wasn't DRY but that worked. Another hurdle was in figuring the intermittent segmentation faults that I found when I tested with Valgrind. I could see that the ant would be missing at the end of some runs and it was clear that that meant that it had gone off the board and then would never return after that. Using Valgrind I was able to see where in the number of steps it had happened. I guessed that it was in my edge checking and maybe that I was missing something in my though process about corners, but no it was an off by one error that was baked into my edge checking and once I used sizeX - 1 and sizeY - 1 it all became clean as a whistle.

I am not sure how well my menu can be reused, but I hope that at least the logical skeleton will be reusable. The while loop for input validation should be able to be used going forward, and while I never used it in 161, the while loop for menus and validation is the first light-bulb that comes to my mind from the reading in that class. I found that implementing the automatic ant positioning was easy to implement once everything else was built out. While I know that seeding rand with srand and using the ctime for the changing input was mentioned in the book, it was quicker to look up an implementation on cplusplus and to just cite the code than it was to find the section in question in the book.