**Design**

Langton’s Ant

Turn a direction and move forward.

White = right turn change space to black

Black = left turn change space to white

User inputs 3x3 Table. Need to compensate for borders. Plus 2 rows and plus 2 columns.

User input will be added by +1 x and +1 y. To compensate for borders.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ---- | ----- | ----- | ----- | ----- |
| | | 1,1 | 1,2 | 1,3 | | |
| | | 2,1 | 2,2 | 2,3 | | |
| | | 3,3 | 3,3 | 3,3 | | |
| ----- | ----- | ---- | ----- | ---- |

Up direction moving to the right would +1 y coord. If border turn right.

Up direction moving to the left would -1 y coord. If border turn left.

Right direction moving to the right would be -1 x coord. If border turn down.

Right direction moving to the left would be +1 x coord. If border turn up.

Down direction moving to the right would be -1 y coord. If border turn left.

Down direction moving to the left would be +1 y coord. If border turn right.

Left direction moving to the right would be +1 x coord. If border turn up.

Left direction moving to the left would be -1 x coord. If border turn down.

For loop to take in steps value to keep looping till the amount of steps are done.

Take in values. Rows, columns, steps, starting row, column.

Starting color is ‘ ‘ and starting direction is “up”

Do loops for input validation store values in variables. Then call class object with values to initialize movement.

If (direction “up” && color == ‘ ‘) If current color is white

If (board[x][y+1]) == ‘ ‘ if upcoming color is also white

Color = ‘ ‘ change current color to ‘ ‘ to keep tabs of current space color scheme.

Board[x][y] = ‘#’ change current space that is white to black

Board[x][y+1] = ‘\*’ change upcoming space to where the ant is located

Y = y+1 to compensate for the change of location

If (board[x][y+1]) == ‘ ‘ if upcoming color is black

Color = ‘#‘ change current color to ‘#‘ to keep tabs of current space color scheme.

Board[x][y] = ‘#’ change current color to black since we started with white.

Board[x][y+1] = ‘\*’ to indicate location of ant.

Y = y+1 to compensate for the change of location

If does not equal to any of those

Direction = “down”

**Test Table**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Start | Default Direction | Steps | What is supposed to happen | What happened | What I changed |
| Middle | Up | 3 | Turn right and change previous color to black and continue to go until it makes a square of black characters | Turn right and change previous color to black but then left remnants of \* in weird places | Changed if statements into if and else if statements so if statements won’t get repeated by accident when it was supposed to end the turn |
| Middle | Up | 10 | Turn right and change previous color to black and continue to go making a square as well as changing blacks into whites. | Changed colors on the right of the up direction however direction did not change often. | Inserted direction changes into the if and else statements |
| Border | Up | 3 | If it sees a border it changes directions based on which direction the border was so it can move forward while skipping a step. | Skipped several steps instead of just one. | Changed the logic of the direction of the ant if it approaches a border. If ant is turning to a border change direction so if it goes the same direction it goes into a space. |
| A value bigger than the array itself | N/A | N/A | Should stop at the starting and keep asking for values until it is appropriate for the amount of space when creating the array | Did not loop and created an error | Changed logic instead of && for the do while changed to ||. |
| Middle | Up | 36 + a restart | Should run smoothly printing each board with the ant moving. Deletes the original board then starts again | Ran smoothly the first run however, the second run created a segmentation error. | Changed the location of where I deallocated memory within the function that it was created in. |
| A string value instead of a integer | N/A | N/A | Should keep running the question to keep asking the user to input a valid number that was listed to the user | Looped properly until inputted appropriate number that then exited the loop | No changes were done. |

**Reflection**

During this process I knew I had to go about solving the logic problems first since those took the longest. I drew out a lot of my design on a notebook piece of paper that was quite messy so that is why I translated most of it into this document. Initially I came up with a loop that used a for loop and if loops that changed the next square to the ant itself however, I came into the issue that I didn’t know how to keep the current color that the ant was standing on since we weren’t showing that the color changed underneath the ant itself but only showing the ant. I changed the if statements to include a color parameter which since it always starts on white. It takes into that consideration and changes this local function variable into whatever the color is suppose to be under the ant. Since it keeps the variable color after the ant moves it can then change to that color and changes the variable based on whatever color the ant is on. Another issue came about was that the ant was not functioning how it was supposed to be. I realized it was because I only had if statements that caused the program to go through each statement without skipping and since there were changes in color and such, the program started to do wonky things and had several ants on the board while not going properly anywhere. After the logic was down solid I came into the issue of segmentation errors which I realized it was because the program wanted me to return something since I had string functions instead of void functions which edit the array and prints out the board without returning anything. After I had solved the segmentation problem I then came about another issue in which my menu function was not really a function and just my main function. I separated the do while loops asking the user to input values into variables that would be used for the ant object into a separate file that would be a menu and have the start of the program be in the main function. Overall this is a lot more coding then what I was used to in 161 however, I do enjoy the fact that we have the freedom to do whatever we want to solve a problem. I had troubles with the final programs that 161 had since it forced you to do certain things which I could not wrap my mind around.