Project 1
Yucen Cao

Design description:

In this project, we will implement a program that simulates Langton's Ant. The rule of Langton's Ant is very simple: the ant is placed onto the board that is filled with white spaces, and starts moving forward. For each step forward, the Langton's ant will follow **2 rules**:

- If the ant is on a **white space**, turn **right** 90 degrees and change the **space to black**.
- If the ant is on a **black space**, turn **left** 90 degrees and change the **space to white.**

After that the ant moves to the next step and continue moving forward. The ant will follow these rules, and continue moving around the board, until the number of steps runs out.

Design for my program:

Board class: No board class, I use a dynamic int array for the board. 0 will be space , and 1 will be "#" character. In this way, I can print the board every step.

Ant initial direction: The initial direction for the ant is fixed facing up.

Largest number of steps/rows/*columns*: The largest number of steps is 1000. Because we need to print out every step, so I think 1000 steps are enough for the project. And 1000 is much more enough for rows and columns.

Edge cases: Skip the step forward step, turn right and then continue on. In the edge cases, only turn right, the color will not change because the ant doesn't step forward.

Random starting location: I have random starting location in the program. Users can choose enter by themselves or get random location.

Description for my program:

My program has 5 important parts as following:

main: the main function

input_validation: to check the input is validation int or not

menu: to check start it or not, and prints menu options to the screen for the user. Verify the user's input, and return the value back to the program by storing it in an int array.

ant: ant class. In the class, the ant have row, column and orientation member. And it has function to get and change this member.

langtons_ant: the important function to process the logic of langton ant. Print every step.

Ant class:

Orientation for the ant: int ant_orientation;

Ant_orientation will be 0 facing up, 1 facing right, 2 facing down, 3 facing left

Input:

Store the input from user: string temp;

I use temp.clear() to reset the temp, so I can reuse it for input.

Store the input: int input[5];

input[0]:The number of rows for the board.

input[1]:The number of columns for the board.

input[2]:The number of steps during simulation.

input[3]:The starting row of the ant.

input[4]:The starting column of the ant.

Test table:

Test Case	Input Values	Driver Functions	Expected Outcomes	Observed Outcomes

Input float number	1.5	menu_start(temp);
		menu_positive_number(temp);
		int menu_location(string temp, int board)
			Show enter wrong number, please enter again	Show enter wrong number, please enter again

Input character	A or bb or #	menu_start(temp);
		menu_positive_number(temp);
		int menu_location(string temp, int board)
			Show enter wrong number, please enter again	Show enter wrong number, please enter again

Input negative number	-12	menu_start(temp);
		menu_positive_number(temp);
		int menu_location(string temp, int board)
			Show enter wrong number, please enter again	Show enter wrong number, please enter again

Input 0	0	menu_positive_number(temp); int menu_location(string temp, int board)	Show enter wrong number, please enter again	Show enter wrong number, please enter again
Input start location outside	Row = 9; Columns = 9; Start row for ant = 10;	menu_location(string temp, int board)	Show enter wrong number, please enter again	Show enter wrong number, please enter again
Input large steps	Step = 1001	int menu_positive_number(string temp)	Show enter wrong number, please enter again	Show enter wrong number, please enter again
Input start location in the corner and will hit the corner	Row = 9; Columns = 9; Start_row = 1; Start_column = 1;	void langtons_ant(int **matrix, int *input)	The first step will not move, and still be row 1, column 1, and turn right	The first step will not move, and still be row 1, column 1, and turn right

Reflection:

When I write the code, I change the edge case design. At first time, I have some problem for the edge case design.

Then I make this change:

1.Check it is edge case or not

2.If it is edge case, only change orientation right

3. If it is not, step forward, change color, change orientation

From this project I learn a lot for time management. It always doesn't have enough time.

I also get how to check the input validation, I use string for user's input. After that, I will check the items in the string is the right input or not.