Emmet Cooke CS162-400

Langston's Ant Project Plan

Project Design

In the main function, we first ask the user if they want to run the Langston's Ant program. If they do not, we immediately exit the program. No more questions are required. If they do, we ask them if they want the ant's starting point to be random. If so, we randomize the starting position and then ask them for the board size and number of steps. If not, we ask them for the starting position in addition to the other information. From this information, we run the program.

The project has three classes, a square class, a board class, and an ant class.

The square class is the base for the game board, as each space in the matrix is made up of one square. Because of this, it is a simple class with only one variable. An enum is included with this class so that we can easily change the color of the square.

The Board class is the base for the ant class. These are separated so that the distinct values relating to the ant itself can be maintained within its own class. The Board class contains the array of pointers and a variety of functions that allow us to manipulate their values. Finally, there is an overridden function that allows the ant class to print the board instead of the board class itself, as the board class does not know where the ant current is, while the ant class does.

The ant class extends the Board class and contains all our information for the ant itself. Included within the class is a enum for direction, which allows us to easily change the direction of the ant without checking and changing strings. The ant is our main initializer and all the values we ask the user for will be passed to ant. There are some functions that allow us to manipulate the ant's values, but the most important function is one that will be called runAnt.

What runAnt should do is loop through a counter until the number of steps provided by the user is reached. The direction of the ant is checked and then the ant mores in that direction before checking the color of the square it stepped into. At this point the board will be printed, the direction will be changed to follow the directions of the Langston's Ant specifications, and the relevant squares will be changed. When the ant reaches the edge of the board, it will flip to the other side and continue in the correct direction.

This should encompass the full specifications of Langston's Ant.

Pseudocode

Main.cpp

Ask the user if they would like the ant to start in a random spot

Initialize a menu object to output the random question prompt

Take their response into string variable and hold it for later

Initialize a menu object to ask the user if they want to run langston's ant

If they choose to start langston's ant

Langston's Ant Project Plan

If they chose a random starting point

Ask the user how many rows there are

Ask the user how many columns there are

Ask the user how many steps the ant takes

Randomize the ants starting row

Randomize the ants starting column

Initialize an Ant object with the above values

Run the ant

If they did not choose a random starting point

Ask the user how many rows there are

Ask the user how many columns there are

Ask the user how many steps the ant takes

Ask the user for the ants starting row

Ask the user for the ants starting column

Initialize an Ant object with the above values

Run the ant

Else

Quit the program

Square.hpp

Make a Square class

Have an enum for the color of the square, either white, black, or ant

A private variable to hold the squares currentColor

Square(); A default constructor

~Square(); A default destructor

color getColor(); returns the squares current color

void setColor(color); sets the currentColor variable to the color passed in

Square.cpp is not required as all functions will have inline definitions

Emmet Cooke CS162-400

Langston's Ant Project Plan

Board.hpp

Make a Board class

A private variable to hold the number of columns

A private variable to hold the number of rows

A private 2d dynamic array to hold the game board

Board(int, int); Default constructor that takes in 2 ints for columns and rows

~Board(); Default constructor

Square** getGameBoard(); returns the gameboard

Int getColumns(); returns the columns

Int getRows(); returns the rows

Void setColumns(int); sets the column variable to the passed value

Void setRows(int); sets the row variable to the passed value

Virtual void printGameBoard() = 0; overridden printing function for Ant class

Board.cpp

Define the constructor

Use setColumns(int) to set the columns value

Use setRows(int) to set the rows value

Initialize the gameboard to an array of pointers

Outer for loop to initialize the rows

Inner for loop to initialize the columns

Define the destructor

Loop to delete the columns arrays

Delete the rows array

Ant.hpp

Enum for direction (up, down, left, right)

Private variable for direction, which is initialized to up

Private int variable for steps

Langston's Ant Project Plan

Private variable for startingRow

Private variable for startingColumn

Default contructor that takes in 5 ints for number of columns and rows for the board, the starting column and row for the ant, and the number of steps.

Default destructor

Void printGameBoard() override; the overridden function from Board class

Direction getDirection(); gets the current direction of the ant

Void runAnt(); Our main function for running the ant

Void setStartingRow(int); sets the row position of the ant

Void setStartingColumn(int); sets the column position of the ant

Void setDirection(direction); sets the direction of the ant

Ant.cpp

Initialize the default constructor with 5 int values

Pass num columns and num rows to Board constructor

set the starting column of the ant

set the starting row of the ant

set the number of steps the ant needs to take

runAnt function.

Declare three variables to track the current row, current column, and current step

Get the gameboard from the getGameBoard() function from Board class

Create a variable to hold the current squares color while the ant is printed on it

Use a while loop to go through until the declared number of steps is reached

Use a pointer to view the correct interior array

Declare an int to hold the last column position and last row position

Set the color of the square the ant is on to ant

Print the game board

If the direction is up

If the currentRow value is 0, jump to the otherside of the board

Otherwise, decrease it by 1

Reset the pointer to the correct array

Find the color of that square and change it to black if white/white if black

Change the direction of the ant based upon the original color of the square

If the direction is right

If the currentColumn value is the size of the array, jump to the otherside of the board

Otherwise, increase it by 1

Reset the pointer to the correct array

Find the color of that square and change it to black if white/white if black

Change the direction of the ant based upon the original color of the square

If the direction is down

If the currentRow value is the size of the array, jump to the otherside of the board

Otherwise, it increase by 1

Reset the pointer to the correct array

Find the color of that square and change it to black if white/white if black

Change the direction of the ant based upon the original color of the square

If the direction is left

If the currentColumn value is 0, jump to the otherside of the board

Otherwise, decrease it by 1

Reset the pointer to the correct array

Find the color of that square and change it to black if white/white if black

Change the direction of the ant based upon the original color of the square

Langston's Ant Project Plan

Increment the counter.

printGameBoard() function

for loop to go through the outer loop

for loop to through interior loop

if the square's color is white

print a space

if the square's color is black

print a #

if the square's color is ant

print a @

Test Cases

Test Case	Expected Output	Actual Output
Not random Steps > 11,000	The ant starts to do a highway pattern	It takes about 11,000 steps to do the highway pattern, but it did it
Random Steps = 0	The program shouldn't run	The program didn't run
Random Steps > 0	The ant begins in a random location and does a normal pattern	The ant began in a random location and ran through the correct steps

Reflection

My original runAnt() function was a failure. While it completed the first few steps of Langston's Ant correctly, when it stepped onto the first black square, it was unable to discern which was the correct direction to go. After some debugging, I found that what was happening was that I was keeping track of the color of the square that the ant was on, but not the square it was about to step on. Thus, it ended falling into a repeating pattern that spanned only two squares, which was not the desired result.

My second attempt at runAnt() was much more successful. The revamp to the first was to use a set of three pointers: one for the square the ant was on, one for the square the ant is currently on, and one of the square that the ant will be on. This turned out to be much more successful, as I could correctly change the color of the square that the ant had been on and I was able to see the color of the square in front of the ant, so that I was able to correctly set the ants direction.

Otherwise, my design worked as I thought it would. Using a Square class allowed me to have the color of the square easily accessible and the Ant and Board class both worked well to print the board with the ant in the correct position.

Langston's Ant Project Plan

It took a bit to get the runAnt() function working correctly, but after I found out where the main issues lay, I was able to get it working within a short amount of time.