Project 1 (Langston’s Ant) Oliver Solorzano

CS 162

Design

1. Gameboard/Starting game
    a. Ask user for number of rows
    b. Ask user for number of columns
    c. Dynamically create array
        i. Take value and save max for positional limits later
    d. Have user choose number of steps
        i. Will loop through movement until max steps reached
        ii. Set min to one step, no “0”, or max “1000”
    e. User chooses starting position with max values set by array size
    f. Print borders at max array locations
        i. Set outskirts to border by default, change if ant moves near
        ii. Set default values for board spaces (blank)
    g. Run simulation until max steps reached
        i. Print each board each step

2. Ant
    a. If on white space, turn right 90 degrees
        i. Change space to black
    b. If on black space, turn left 90 degrees
        i. Change space to white
    c. Create array that updates ant’s position after every move
        i. That position in array should be updated to “*”
    d. Create variable that updates ant’s orientation after every move
        i. Have one condition for each space ant can land (blank or black)
            1. For each, have four different orientations
                a. N move up one in array
                b. S move down one in array
                c. E move right one in array
                d. W move left one in array
            2. Based on the current orientation, will shift to specific type based on landing conditions
    e. If ant moves to edge, reverse movement to turn it around (or reorient w/o moving)
        i. Modify array values if landing on extreme array positions
3. Menu (switch statement)
    a. Start by asking user to either being program or quit
        i. If quitting, return “0”
    b. If running, proceed to ask for input
        i. Number of board rows
        ii. Number of board columns
        iii. Number of total steps
        iv. Starting row
        v. Starting column

   c. After run is over, ask user if they want to run again. If not, exit.
      i. Loop
4. Input Validation
   a. User should only be entering integers (unsigned)
      i. Check for unsigned int when entered
   b. Should have min and max size for board (min [2][2], max [100][100])
   c. Inform user of limits
      i. If limits breached loop to return for input
   d. Based on board size, ant movement will be limited

Test Table

Test Case	Input	Expected	Observed
User enters valid option for menu	Enters int value within range of menu	Option selected successfully, progresses to next input	Option selected successfully, progresses to next input
User enters invalid option for menu	Enters value != int, or not within range	Tells user entry is invalid, tries again	Tells user entry is invalid, tries again
User enters array size too small or too large	Array smaller than [1][1] or greater than [100][100]	Tells user that value is invalid, tries again	Tells user that value is invalid, tries again
User enters array size within limit	Array size between [1][1] and [100][100]	Array dynamically allocates to proper size with borders set, default blank spaces	Array dynamically allocates to proper size with borders set, default blank spaces
Number of steps too small or too large	"0" number of steps or extreme like "99999999999999"	Tells user value is invalid, please enter within range, try again	Tells user value is invalid, please enter within range, try again
Number of steps within range	"1" -> "1000" steps entered	Program proceeds to run for number of steps, printing each one	Program proceeds to run for number of steps, printing each one
User enters starting position for and or chooses to randomize	Array position set within max limits	Ant begins facing north at specified location	Ant begins facing north at specified location
User enters invalid starting position	Array position set outside max limits	Tell user that value is out of range, try again	Tell user that value is out of range, try again
Ant movement reaches border	Input beyond array size	Reverse ant movement back towards board, or change orientation without moving	Reverse ant movement back towards board, or change orientation without moving

Oliver Solorzano
CS 162

Reflection

This biggest obstacles I came across when writing this program and implementing my ideas, were how to update the spaces the ant landed on, and what to do when the ant reached the edges of the board. When it came to updating spaces, I found that I had no way to store what the character the space would change to after the ant moved. I was initially just tracking the current ant position and updating that on the board. I had if statements to determine what the spaces would change to, but the ant would change the space to '*'. I decided to create a separate set of variables that would store the space character while the ant was still occupying that space. When the ant would move, the space would be updated to character stored in that variable. Initially, I wanted to create a separate array that tracked the ant's movement, I would then check that after every move to know when the ant was close to the border. This solution became increasingly complicated, and the mixing of the board and position arrays created a messy program. After realizing that I could just use the borders as indicators, I switched my design. I was already having to program look ahead to see what kind of space the ant would land on, and depending on whether or not it was blank or black, the ant would move accordingly. I expanded on this idea, and added the border characters as an additional step in this if statement branch. The program would then check for a space, blank, or border character, and if it came across a border character, the ant would not move, only change its direction to the opposite orientation. It was this change that made me realize how important simplicity is to writing good code. I was initially trying to implement a complex code, when the solution was stupid-simple. I learned to seek the simple solutions, because the end result is the same (and it makes debugging a lot less frustrating).