COMP281 2020 Assignment 1 Report

1081: Game of Life

I decided to create a two-dimensional array to represent the board and called it “board”. This is declared after the user inputs how many rows and columns it should have and is subsequently filled with the user’s start configuration.

To perform a step, the program first creates a new two-dimensional array with the same dimensions as the “board” array. I called this array “newBoard” as it would hold the configuration of the old board but with the appropriate changes. It is sequentially filled with the new status of each square from the old board based on its neighbours. At the end of every step the old board is updated by replacing each square with its corresponding square in the “newBoard”.

To work out the new status of a square I identified adjacent squares and counted, out of these, how many were alive (ignoring squares that were outside of the board). Some if statements based on the game of life rules in the specification then worked out what the new status should be.

I found adjacent squares by creating a global two-dimensional array called “neighbours” which stored 8 pairs of values to represent a row and column number. The function “findNeighbours” takes a column and row number of a square and fills the “neighbours” array with the column and row numbers of every adjacent square. The column and row number of the adjacent squares were found using the principle that for any square with column and row numbers (x,y), the surrounding squares would have the following column and row numbers:

x-1, y-1	x, y-1	x+1, y-1
x-1, y	x, y	x+1, y
x-1, y+1	x, y+1	x+1, y+1

NOTES

- (May seem counter intuitive because it isn’t based on co-ordinates in a graph but rather the columns and rows in my two-dimensional array. Although, I did try to keep the values in a co-ordinate relating to the x and y axis hence why the (x,y) refers to (column,row))
- (The order of the pairs in the “neighbours” array was arbitrary since I only cared about their actual co-ordinates.)

The number of steps is dependent on the users input as detailed in the specification.

1086: Run Length Encoding of ASCII Art

Unfortunately, I couldn’t get my program to work in the way that was specified. My program seems only capable of compressing and expanding single lines (which really annoyed me because I don’t understand why). In Online Judge I got an output limit exceed which I assume is because the conditions to break out of my loops if an EOF is detected were not triggered. I hope I can pick up some decent marks as I thought the most interesting parts of this task were the algorithms to compress and expand input text which I did get working.

Main

This function just scans the first letter in the input file and uses this to determine whether the user wants to compress or expand the subsequent input text. Afterwards it makes the appropriate function call.

Compressing Text

To compress text, I set up a loop that broke at the end of the file after outputting the final compressed text. At each step the next character input was stored in a variable called “nextCHAR”. Given that the “nextChar” is not the end of the input file, the program checks whether it matches the “currentCHAR”. “currentCHAR” stores the character that is being currently tracked. If they match, then a count is incremented. If not, the program makes an appropriate output. If the run count was greater than one then the program makes the following output: 2 of the current character followed by an integer representing the run count followed by an asterisk. Otherwise, the program only outputs the “currentCHAR”. Immediately after the output, the “currentCHAR” is set to whatever the “nextCHAR” is

(Which at this point is the last thing that was scanned) and the count is reset to 1. This is so that in the next step, the "currentCHAR" is the new character to be tracked until a different character is scanned in.

If the program reaches the end of the input file, it makes the last output using the same steps as the algorithm described above before breaking out of the loop.

Expansion

To expand text, I set up a loop that broke at the end of the input file. At each step, the program scanned a new character and stored it in "currentCHAR". If the "currentCHAR" is not the same as the "previousCHAR" (the variable storing the character read before the current one) then the program outputs the current char and sets the "previousCHAR" to the "currentCHAR". If they are equal, then it means there must be a run of that character in the uncompressed text. So, the program knows the next character is an integer and uses this to output the appropriate number of characters in the run. It does this using a for loop which iterates "i" times where "i" is the integer read minus 1 (since the program only detects a run after outputting the first character in the run).