Introduction

Discussion

I have used a variety of data structures to build this application, including linked lists, stacks, a recursive method, and a method that is identical to one that would be used to do a depth first search through a binary(or otherwise) tree.

Before I get ahead of myself, let me discuss the classes used in this program. There is a Singly Linked List class, as well as a SLL Node class, which simply provide linked list capabilities to the application. The Stack class utilises these SLL classes to create a stack of Block objects. There is also a Coordinate container class that merely consolidates two integer values into one object. This leads us directly to the Block class. The Block class can be regarded as a LL Node, as it contains references to four other Block objects, namely n, w, s, & e. These references represent the adjacent blocks. The block also contains the blocks coordinates, and a number of aptly named Boolean variables. These are “visited”, which will be explained a little bit later, “nWall”, “wWall”, “sWall”, & ”eWall”, which represent the “walls” which exist in each block. These variables are used in generating the maze to be solved. “empty”, and “solution” are also Boolean, and these are used in determining a solution to the maze. Block also has a conviniece method called “hasUnvisitedNeighbours()” which returns true if none of the Blocks adjacent blocks has the variable “visited” set to true.

Conclusion