Approach Rationale

I chose to use a 2D array implementation over a linked list approach for the following reasons:

Simpler implementation

2D arrays are ideal for storing data with fixed dimensions since its size is predefined whereas linked lists are ideal for projects where the frequent inserting, deleting, and resizing of elements is required.

Since this project requires me to read/access and update values from a grid of fixed dimensions, the 2D array is a better choice.

The 2D array easily allows me to iterate through the indexes to read and update values whereas using a linked list would have required me to work with many different pointers thereby increasing the complexity of memory management.

The 2D array also allows for more simple indexing using (map[i][j]) to access data at an index whereas a linked list would require more complex code to go through the list and find the required position/data.

Class Design

Class grid

This class was designed to hold the functions responsible for manipulating the map that the robot is traversing through in terms of adding O/G, outputting the next move, clearing the map as well as calculating the potentials.

Public variables and functions:

- **void add_g_o(int x, int y, char T); //function to add goal or obstacle to position within array**
  Parameters: int x, int y, char T
  Return Type: n/a
  Rationale for design: It begins by checking if the position is within the map, if not in bounds then you cannot add a goal or obstacle and the function will output failure. If it is within bounds, the functions add a char O/G to this position using the add_description() function based on input from the user for char T.
- **void update_k(float new_k); //function to update k value**
  Parameters: float new_k
  Return Type:n/a
  Rationale for design: It begins by checking if the k value is valid, if yes then it updates it then begins to iterate through the entire map to recalculate potentials. It calls on the get_description() function to see if the position is a O/G and then calls the calculate_potential() function, outputs success at the end. Output is failure if the k value is not valid, also skips updating k value.
- **void clear(); //function to clear a grid of goals and obstacles**
  Parameters:n/a
  Return Type:n/a

Rationale for design: It begins by checking if a map has been created, if yes it continues by iterating through each cell and setting the potential to 0. If no map has been created, there is nothing to set to 0. Success or failure are output accordingly.

- **void next_move(int x, int y);**
  Parameters: int x, int y
  Return Type:n/a
  Rationale for design:Function checks if location is within the map and then calls get_potential to output the value of the potential at the position.

Private variables and functions:

- **void calculate_potential(int xg, int yg, char T)//function calculates potential for map**
  Parameters: int xg, int yg, char T
  Return Type: n/a
  Rationale for design: Function iterates through map and first checks if robot is in same position as obstacle or goal, if yes then skips calculating potential for this position. Then it calls the function calculate_potential_cell() to calculate for the exact position, it also calls on get_potential() to get the current potential of the cell, then it uses the add_potential() function to update the value of its potential.
- **float calculate_potential_cell(int x, int y, int xg, int yg, char T)//calculate potential for a cell**
  Parameters:int x, int y, int xg, int yg, char T
  Return Type: float
  Rationale for design: Function checks if we are in the same position as obstacle or goal, if yes then potential is set to 0. If not then proceeds with formula calculation and then checks if it is a goal, if yes it multiplies by -1

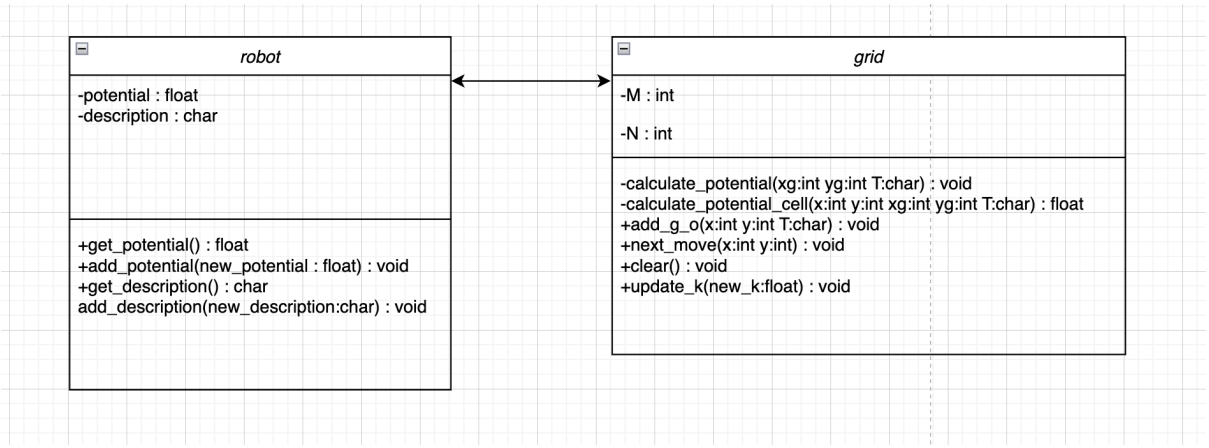
Class robot

This class was created to hold the functions that read and manipulate the O/G descriptions of the cells as well as the potential values, which will be used by grid class in its functions to access data.

Public variables and functions:

- **void add_potential(float new_potential);**
- **void add_description(char new_description);**
  Both functions above will update the potential or description values for a cell respectively

UML Diagram


robot
-potential : float
-description : char
+get_potential() : float
+add_potential(new_potential : float) : void
+get_description() : char
add_description(new_description:char) : void
grid
-M : int
-N : int
-calculate_potential(xg:int yg:int T:char) : void
-calculate_potential_cell(x:int y:int xg:int yg:int T:char) : float
+add_g_o(x:int y:int T:char) : void
+next_move(x:int y:int) : void
+clear() : void
+update_k(new_k:float) : void