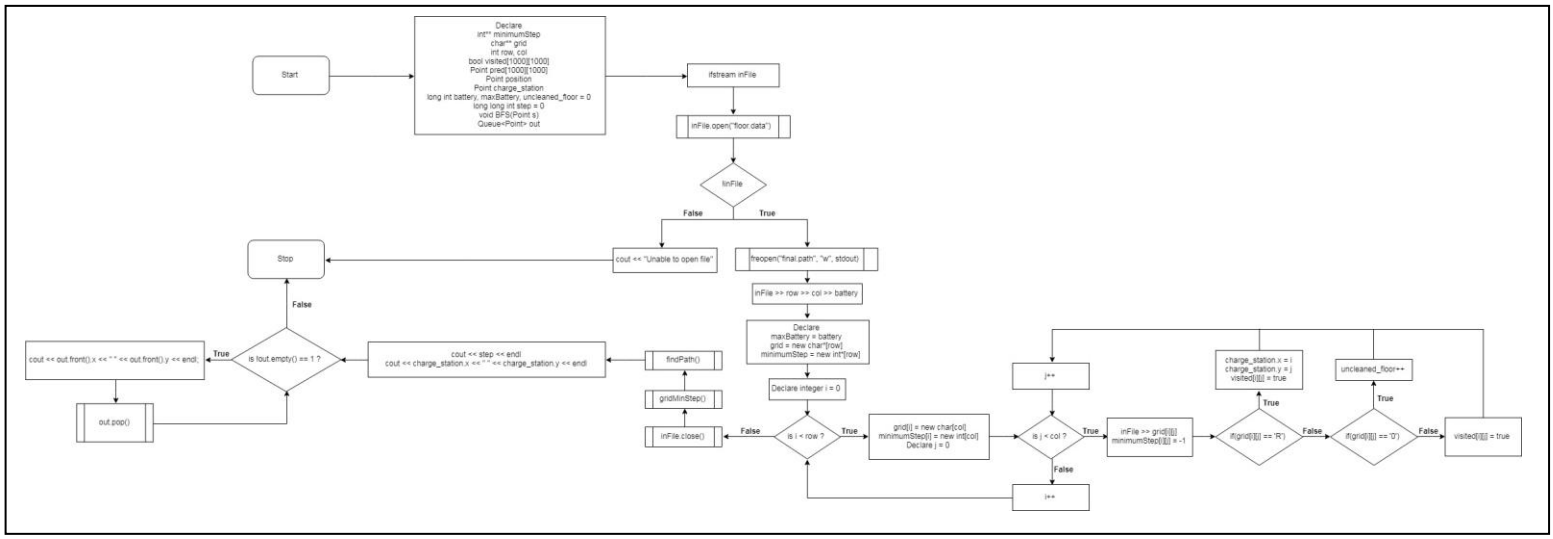
Data Structure

Project 2 – Floor Cleaning Robot

學號:107062361 姓名:許珉濠

1. Project Description

1.1. Program Flow Chart



1.2. Detailed Description

In the first rectangle after start process is an process of declaring global variable. The purpose of declaring global variable is to avoid passing variable to function which can provoke mistake in writing code. Afterwards, **ifstream inFile** is declared to read input file. The way to read a input file can be done by calling **inFile.open("**filename**")** function. In this case, the file that is being read is "**floor.data**". If the **inFile**'s value is zero after read input file, it means that the read input file is failed. Therefore, the program will immediately close down. On the occasion that read input file is successful, function **freopen("final.path", "w", stdout)** will be called. The purpose of calling this function is instruct the program to write all the program output into "**final.path**" file.

Since the input source of the program is from input file, it needs to use **inFile** as the input stream. The first two variable of the input is consists of map's row and column, and the third one is robot initial battery as well as the maximum battery. The battery will be stored inside **maxBattery** to keep the maximum battery value due to recharge feature is available. For loop will be used to build the map and minimum step from battery charge station to every map point with **row** and **col** size. The map input values is stored

in 2D char variable **grid** and the minimum step is stored in 2D integer variable **minimumStep**.

First, all the minimumStep will be assigned as -1. Consider that the map input value is '**R**', it will be the point of battery charge station. If the value is '**1**', it will be the wall in the map. However, if the map input value is '**0**', it is the floor that is needed to be cleaned by the robot. At the same time, 2D boolean variable **visited** which will be used in BFS is assigned as true if the map input value are either '**R**' or '**1**'. If the map input value is '**0**', then it is assigned with false.

Subsequently, read from input file is done. The input stream from file **inFile** is stopped with function **inFile.close()**. Then, the minimum step of every point in map can be found by calling function **gridMinStep()**. In this function, it uses Queue **que** with class type **Point** as base to do BFS. **Queue** is made from class since STL is prohibited to be used. On the other hand, **Point** is a class stores two integer value which is coordinate **x** and **y**. The function starts do 4 directions BFS with battery charge station as starting point. While doing BFS, it will check whether the next point is already visited and the next point doesn't surpass the map. If the condition is true, then the next point will be pushed to the **Queue**. The minimum step of those point will increment by one from current point minimum step. It also store the predecessor of next point into 2D integer array **pred**.

After all the minimum step is found, the floor cleaning will be started by finding the path. I use for loop to check whether every point in the map is already visited or apparently it is a wall. If the point is floor, then it will be locked as target cleaning. The robot will go to there by using **gotoPath** function. In **gotoPath** fuction, it will use **pred** to find the way from either source to destination or destination to source and push every **pred** to Queue **que** with class type **Point** repeatedly with while loop. When the destination and source point have met, the loop will stop. It start print **que** with function **printQue**. In these function, it push the back of parameter Queue **q** into Queue **out**, decrement the battery by one and increment **step** value by one. In addition, the **visited** and **grid** of this point is set into **true** and '**2**' respectively. Assume that if the pushed point is battery charge station point, then the battery will be recharged with **recharged** function. This process keep repeatedly done until the **q** is empty. The **out** is Queue to store the robot path which will be printed in the end after all floor is cleaned.

If **gotoPath** function ends, it will continue call **BFS** function. This function is made to command the robot randomly clean the nearby uncleaned floor by using 4 direction BFS if the battery is enough. If any path is found, then it will push the next point and do push point to **out**, assign visited point to true, etc which is similar to **gotoPath**. If the battery is not enough, the robot will be commanded to back to battery charge station. Thus, the process is done repeatedly until all the floor is cleaned.

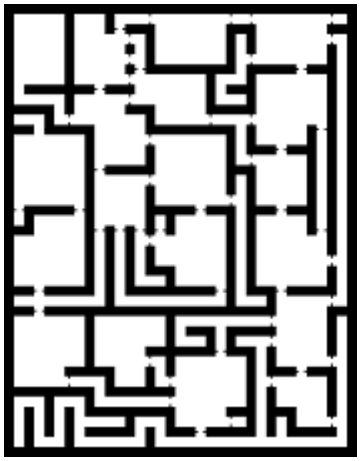
Thereupon, the rest things to do is just simply print out **step** value, the battery charge station point, and all the path in **out**.

2. Test Case Design

2.1. Detailed Description of the Test Case

The test case map size is **226 x 176** with **1100** as initial battery value. The method that I use to make the testcase is mapping white-black photo into 0 and 1 value respectively. First, I download a dungeon maze map from the following link.

https://donjon.bin.sh/pathfinder/dungeon/index.cgi



Picture for mapping

Then, the mapping tools can be found in the following link.

https://www.dcode.fr/binary-image

Therefore, this is the simple way I create the test case.