- 程式解說
  - main()

```cpp
int main() {
    char queens[10][10];

    while(cin >> C >> R >> pre_placed_amount){ // input the data
        for(int i=0; i<R; i++) { // initialize the chessboard
            for(int j=0; j<C; j++) {
                queens[i][j] = '.';
            }
        }

        for(int i=0; i<R; i++) { // intialize the answer chessboard
            for(int j=0; j<C; j++) {
                ansQueens[i][j] = '.';
            }
        }

        maxQueens = 0;
        for(int i=0; i<pre_placed_amount; i++) { // input the pre-placed queens
            cin >> pre_placed[i].c >> pre_placed[i].r;
            queens[pre_placed[i].r][pre_placed[i].c] = 'Q';
            ansQueens[pre_placed[i].r][pre_placed[i].c] = 'Q';
        }


        placeQueen(queens, 0); // start to place the queens

        print_ansQueens();
        cout << "Additional Queens amount: " << ansQp_index << endl;
        print_ansQueensPos();
    }


    return 0;
}
```

在主函式中，我們先進行基本的資料輸出，以及開始放入棋子，最後將結果輸出。

- 自定義型別 pos ，存入r, c

- int R, C => 棋牌的大小

- int maxQueens => 紀錄現在最多可以放幾個Queens
- char ansQueens[10][10] => 存最後的棋盤答案

- int pre_placed_amount => 一開始要放的數量
- pos pre_placed[100] => 存入剛開始要放入的座標

- int ansQp_index => 還可多放幾個Queens的數量
- pos ansQueensPos[100] => 存可多放Queens的座標

- checkQueen()

```
bool checkQueen(char queens[10][10], int row, int col){ // Check if this location can be placed
    for(int i=0; i<R; i++) { // check this col
        if(queens[i][col]=='Q' && i != row)
            return false;
    }

    for(int i=0; i<C; i++){ // check this row
        if(queens[row][i]=='Q' && i != col)
            return false;
    }

    for(int i=row-1, j=col+1; i>=0 && j<C; i--, j++) { // Upper right
        if(queens[i][j]=='Q')
            return false;
    }

    for(int i=row-1, j=col-1; i>=0 && j>=0; i--, j--) { // Upper left
        if(queens[i][j]=='Q')
            return false;
    }

    for(int i=row+1, j=col+1; i<R && j<C; i++, j++) { // Lower right
        if(queens[i][j]=='Q')
            return false;
    }

    for(int i=row+1, j=col-1; i<R && j>=0; i++, j--) { // Lower left
        if(queens[i][j]=='Q')
            return false;
    }

    return true; // If it passes all the checks, it can be placed
}
```

檢查這個位置是否可以放，因此我們開始檢查，整個row、整個column，以及左上、右上、左下、右下等位置。

- isPrePlaced()

```
bool isPrePlaced(int r, int c){ // Check if this location is pre-placed
    for(int i = 0; i < pre_placed_amount; i++){
        if(pre_placed[i].r == r && pre_placed[i].c == c){
            return true;
        }
    }

    return false;
}
```

檢查這個座標是不是題目本來要放的位置。

- placeQueen()

```
1  void placeQueen(char queens[10][10], int row) {
2      int QQ = howManyQueens(queens);
3      if(QQ > maxQueens){
4          // printQueen(queense);
5          ansQp_index = 0;
6          maxQueens = QQ;
7          for(int i=0; i<R; i++) {
8              for(int j=0; j<C; j++) {
9                  ansQueens[i][j] = queens[i][j];
10                 if(!isPrePlaced(i, j) && queens[i][j] == 'Q'){
11                     ansQueensPos[ansQp_index].r = i;
12                     ansQueensPos[ansQp_index].c = j;
13                     ansQp_index++;
14                 }
15             }
16         }
17     }
18
19     if(row == R) { // If all rows are placed, return
20         return;
21     }
22
23     bool nexted = false; // whether the next row has been placed
24     for(int j=0; j<C; j++) {
25         // printQueen(queens);
26         if(checkQueen(queens, row, j) && !isPrePlaced(row, j)) {
27             // cout << "row" << row << "j" << j << endl;
28             queens[row][j]='Q'; // Case 1: Place the queen
29             nexted = true; // the next row has been placed
30             placeQueen(queens, row+1);
31             queens[row][j]='.'; // Case 2: Not place the queen
32         }
33     }
34
35     if(!nexted){ // if the next row has not been placed, then place the queen in the next row
36         placeQueen(queens, row+1);
37     }
38 }
```

L2-L17 : 去比較現在的答案是否是最多的可放Queen，如果是最多的就將答案做更新
L19-L21：如果row等於R代表說已經走到最底了，結束此遞迴
L23：紀錄是否有進入下個row的遞迴
L24-L33：在這個row下，將每個col都試過一遍，並且判斷這個位置可不可以放，以及這個位置是不是題目說要預放的，如果可以放，則會有要放即不要放的兩種情形，因此我們先放置一個Queen，並且遞迴下去，遞迴結束後，再將此位置清空，考量下個情形。
L35-L37：如果這個row都不能放了，我們自己進入下個row

- 程式輸出結果

```
5 3 2
0 0
4 2
Q . . . .
. . Q . .
. . . . Q

Additional Queens amount: 1
R C
1 2
```

```
5 3 2
1 1
4 2
. . . Q .
. Q . . .
. . . . Q

Additional Queens amount: 1
R C
0 3
```

```
10 10 2
0 0
8 9
Q . . . . . . . . .
. . . Q . . . . . .
. . . . . Q . . . .
. . . . . . . . . Q
. . . . . . . Q . .
. Q . . . . . . . .
. . . . Q . . . . .
. . Q . . . . . . .
. . . . . Q . . . .
. . . . . . . . Q .

Additional Queens amount: 8
R C
1 3
2 6
3 9
4 7
5 1
6 4
7 2
8 5
```

```
7 5 0
Q . . . . . .
. . Q . . . .
. . . . Q . .
. Q . . . . .
. . . Q . . .

Additional Queens amount: 5
R C
0 0
1 2
2 4
3 1
4 3
```

- 心得

  八皇后是很經典的題目，以前高中有寫過，當初的目的是將所有組合窮舉出來，而這份作業不大一樣的是他還多了先放的皇后，並且輸出多放的位置與數量，但整體要改的東西也不算多，只是因為`return`的問題，卡了一整天，拿出來後再加上`if`就過了，整體寫下來我覺得還算簡單。我整體的解題思路主要是利用遞迴將答案窮舉出來，我覺得可以試試非遞迴的方式來解這題，以效率來說應該會快很多，在寫這題的同時，也教了朋友們如何去解這題，講解整個思路，是個使我受益良多的作業。

- 補充說明

  iLearn上繳交的程式碼是為了符合checker的輸出所改的，而報告上的輸出則是放在GitHub上，歡迎助教與老師參閱，謝謝。

  https://github.com/alecwu44743/c_learning/blob/main/fcu_cs/22fall_Data_Structures/DS_HW1-EightQueens/queen.cpp