Eight Queens

There are 8 possible positions to place the Queen in the first column. For each position in the first column, there are 8 possible positions to place the Queen in the second column.

....

For each position in the seventh column, there are 8 possible positions to place the Queen in the last column.

Thus, there are $8 \times 8 \times 8 \times 8 \times 8 \times 8 \times 8 \times 8$ possible ways in total. $\sim 16.8$ million

We need to set up all these boards to identify which ones are valid.

This approach is called brute force.
Setup board

for ( int a = 0 ; a < 8 ; ++a )
for ( int b = 0 ; b < 8 ; ++b )
for ( int c = 0 ; c < 8 ; ++c )
for ( int d = 0 ; d < 8 ; ++d )
for ( int e = 0 ; e < 8 ; ++e )
for ( int f = 0 ; f < 8 ; ++f )
for ( int g = 0 ; g < 8 ; ++g )
for ( int h = 0 ; h < 8 ; ++h )
{
    //Do something!
}
}
PSEUDO CODE – DO SOMETHING

1. Position our Queens

    board[a][0] = 1;
    board[b][1] = 1;
    board[c][2] = 1;
    board[d][3] = 1;
    board[e][4] = 1;
    board[f][5] = 1;
    board[g][6] = 1;
    board[h][7] = 1;

    Use of board is optional, especially if you can visualize the setup.
PSEUDO CODE – DO SOMETHING

2. Make sure the positioning of the Queens cannot attack each other.

If this is the case, this board passes ‘the test’.

Starting from right most piece:
- Find where the Queens is on the current column
- Make sure there isn’t another Queen on this row
- Make sure there isn’t another Queen in up diagonal direction
- Make sure there isn’t another Queen in down diagonal direction
PSEUDO CODE – DO SOMETHING

3. If a board setup passes the test

Print out the board setup
Keep track how many boards are found
Idea Behind Backtrack

Trial and error approach

- If a position is not suitable then try the next one.
- If none of the position are suitable then back track to previous column and try the next position in that column.
- When a solution is found call backtrack to look for others. (at this point column would be 8)
- When backtrack takes you back to the very beginning (column = 0, row = 7) backtracking again would push column to -1, therefore you found all solutions.
Setup

int b[8][8] = {0};

int counter = 0;

int r;

int c = 0;

b[0][0] = 1;
Working with columns

NC: c++;

if (c == 8)

goto printBoard;  //what goes here?

r = -1;
NR: r++;  
if (r==8)  
goto back;    //what goes here?
for (int i=0; i<c ; i++)  
    if (b[r][i]==1) goto NR;
for (int i=1; (r-i)>=0 && (c-i)>=0 ; i++)  
    if (b[r-i][c-i]==1) goto NR;
for (int i=1; (r+i)<8 && (c-i)>=0 ; i++)  
    if (b[r+i][c-i]==1) goto NR;
b[r][c]=1;  
    //what goes here?
backTrack: c--; 
if ( c == -1 ) 
{ //what goes below? 
  cout << counter << endl;  
  return 0; 
} 

r=0; 
while (b[r][c]!=1) 
  r++; 
  b[r][c]=0; 
  //what goes here?
Printing Board

printBoard: counter++;  
cout << "Solution #"<<counter<<":"<<endl;  
for(int i=0 ; i<8 ; i++)  
{  
    for (int j=0 ; j<8 ; j++)  
        if ( b[i][j] == 1 )  
            cout << "Q";  
        else  
            cout << "-";  
    cout <<endl;  
}  
cout <<endl;  

// what goes here?