```
/*
    File name: hw2.cpp
    Created by: Tan Qi Hao
    Created on: 9/13/2019
    Synopsis: This program play the game called game of life.
*/

#include <iostream>
#include <cstdlib>

using namespace std;


const char ALIVE = '*';
const char DEAD = ' ';

void initialization(bool **world, int nrows, int ncols);
// prompts and reads the alive cells to initialize the world
// initializes the world

void generation(bool **world, bool **copy, int nrows, int ncols);
// input parameters: original world, an array to make a copy, dimensions of the
array
// updates the world

void display(bool **world, int nrows, int ncols);
// prints the world to the console

int main(){

    // Variable declarations. You can add more if necessary
    bool **world, **copy;
    int nrows, ncols;
    char next;

    cout << "Enter world dimensions (rows and columns): ";
    cin >> nrows >> ncols;

    // allocate memory for dynamic 2d-arrays 'world' and 'copy'
    world = new bool*[ncols];
    for(int i = 0; i < ncols; i++){

      world[i] = new bool[nrows];

    }

    world = new bool*[ncols];
    for(int j = 0; j < ncols; j++){

      world[j] = new bool[nrows];

    }

    // initialize the world and display
    initialization(world, nrows, ncols);
    display(world, nrows, ncols);

    // prompt user input, Generation/Quit
    while (true){
        cout << "next Generation or Quit (g/q): ";
      cin >> next;
      if (next=='g' || next=='G' || next=='q' || next=='Q')
          break;
    }

    while (next=='g' || next=='G'){

        // update the world and display
        generation(world, copy, nrows, ncols);
        display(world, nrows, ncols);

      // prompt user input
        while (true){
          cout << "next Generation or Quit (g/q): ";
          cin >> next;
          if (next=='g' || next=='G' || next=='q' || next=='Q') break;
      }
    }

    // deallocate memory for dynamic 2d-arrays 'world' and 'copy'
      for(int i = 0; i < ncols; i++){

      delete [] world[i];

    }
    delete [] world;

    for(int i = 0; i < ncols; i++){

      delete [] copy[i];

    }
    delete [] copy;

    return 0;
}


void generation(bool **world, bool **copy, int nrows, int ncols){

//Copy world array to copy array
  for(int i = 0; i < nrows; i++){

    for(int j = 0; j < ncols; j++){

    copy[i][j] = world[i][j];

    }

  }

 //Determines the life and dead of cells
  int neighborSum = 0;
  for(int k = 0; k < nrows; k++){

    for(int l = 0; l < ncols; l++){

     if(world[k][l]){
      if(world[k-1][l-1]){
      neighborSum++;
      }

      if(world[k][l-1]){
      neighborSum++;
      }

      if(world[k+1][l-1]){
      neighborSum++;
      }

      if(world[k-1][l]){
      neighborSum++;
      }

      if(world[k+1][l]){
      neighborSum++;
      }

      if(world[k-1][l+1]){
      neighborSum++;
      }

      if(world[k][l+1]){
      neighborSum++;
      }

       if(world[k+1][l+1]){
      neighborSum++;
      }

      //Rules when the cells is alive

       if(neighborSum < 2 || neighborSum > 3){
       world[k][l] = false;
       }else{

       world[k][l] = world[k][l];

       }


     }

    }

  }

}


void initialization(bool **world, int nrows, int ncols){
  int numAliveCell;
  int coord1, coord2;

  //initialize all of the boolean world array to false
 for(int i = 0; i < ncols; i++){
    for(int k = 0; k < nrows; k++){

      world[i][k] = false;

    }

  }

  cout << "Enter number of alive cells: ";
  cin >> numAliveCell;

  cout << "Enter coordinates of alive cells: " << endl;

  //Ask for coord;
  for(int j = 0; j < numAliveCell; j++){

    cin >> coord1 >> coord2;

    world[coord2][coord1] = true;
  }

}


void display(bool **world, int nrows, int ncols){
  for(int i = 0; i <= ncols + 1; i++){
    cout << "=";
  } //Top border

  cout << endl;

  for(int j = 0; j < nrows; j++){

    cout << "|"; //Border

    for(int k = 0; k < ncols; k++){
      if(world[k][j]){

      cout << ALIVE;

      }else{

      cout << DEAD;

      }

    }

    cout << "|" << endl;

  }

for(int l = 0; l <= ncols + 1; l++){
    cout << "=";
 } //lower border

 cout << endl;
```

```
}
```