```
#include <iostream>
#include <cstdlib>
#include <ctime>

using std::cout;
using std::cerr;

class Map {
protected:
	int *data;
	int width;
	int height;
	int offset;

	int countHotspots();
	int findHotspotPosition(const int hotspotID);
	void drawLine(const int position, const int direction);
	void drawBorder();
	void generateHotspots();
	void sanitizeHotspots();

public:
	void print();
	void generate(const int pillarCount);
	bool init(const int width, const int height, const int offset);
	void deinit();
	Map(){ data = NULL; width = 0; height = 0; }
	~Map() { deinit(); }
};

int main(int argc, char *argv[]) {
	srand(unsigned(time(NULL)));

	Map map;
	if (not map.init(61, 28, 2)) return 1;

	map.generate(4);
	map.print();

	return 0;
}

int Map::countHotspots() {
	int cnt = 0;

	for (int y = offset + 1; y < height - 1; y += (offset + 1)) {
		for (int x = offset + 1; x < width - 1; x += (offset + 1)) {
			if (data[y * width + x] == 2) cnt++;
		}
	}

	return cnt;
}
```

```
int Map::findHotspotPosition(const int hotspotID) {
	int cnt = 0;

	for (int y = offset + 1; y < height - 1; y += (offset + 1)) {
		for (int x = offset + 1; x < width - 1; x += (offset + 1)) {
			if (data[y * width + x] == 2) {
				if (cnt == hotspotID) return y * width + x;
				else cnt++;
			}
		}
	}

	return 0;
}

void Map::drawLine(const int position, const int direction) {
	int pos = position;

	while (data[pos] != 1) {
		data[pos] = 1;

		if (direction == 0) pos -= width; // UP
		else if (direction == 1) pos -= 1; // LEFT
		else if (direction == 2) pos += width; // DOWN
		else if (direction == 3) pos += 1; // RIGHT
	}
}

void Map::drawBorder() {
	for (int y = 0; y < height; y++) {
		data[y * width] = 1;
		data[y * width + width - 1] = 1;
	}

	for (int x = 0; x < width; x++) {
		data[x] = 1;
		data[(height - 1) * width + x] = 1;
	}
}

void Map::generateHotspots() {
	for (int y = 0; y < height; y += (offset + 1)) {
		for (int x = 0; x < width; x += (offset + 1)) {
			data[y * width + x] = 2;
		}
	}
}

void Map::sanitizeHotspots() {
	for (int y = 0; y < height; y += (offset + 1)) {
		for (int x = 0; x < width; x += (offset + 1)) {
			if (data[y * width + x] == 2) {
				data[y * width + x] = 1;
			}
		}
	}
}
```

```
void Map::print() {
	for (int y = 0; y < height; y++) {
		for (int x = 0; x < width; x++) {
			int value = data[y * width + x];
			if (value == 0) cout << " ";
			else if (value == 1) cout << "#";
			else if (value == 2) cout << ".";
			else cout << "?";
		}
		cout << "\n";
	}
}

void Map::generate(const int pillarCount) {
	int hotspots = countHotspots();
	while (hotspots > pillarCount) {
		int randHotspotID = rand() % hotspots;
		int direction = rand() % 4;
		int position = findHotspotPosition(randHotspotID);

		drawLine(position, direction);
		hotspots = countHotspots();
	}

	sanitizeHotspots();
}

bool Map::init(const int width, const int height, const int offset) {
	if (((width - 1) % (offset + 1)) != 0) {
		cerr << "ERROR: Bad width\n";
		return false;
	}
	if (((height - 1) % (offset + 1)) != 0) {
		cerr << "ERROR: Bad height\n";
		return false;
	}

	Map::width = width;
	Map::height = height;
	Map::offset = offset;

	data = new int[width * height];
	if (data == NULL) return false;

	for (int i = 0; i < width * height; i++) data[i] = 0;

	generateHotspots();
	drawBorder();

	return true;
}

void Map::deinit() {
	if (data != NULL) {
		delete[] data;
		data = NULL;
	}
}
```