Étude de cas : le jeu de la vie

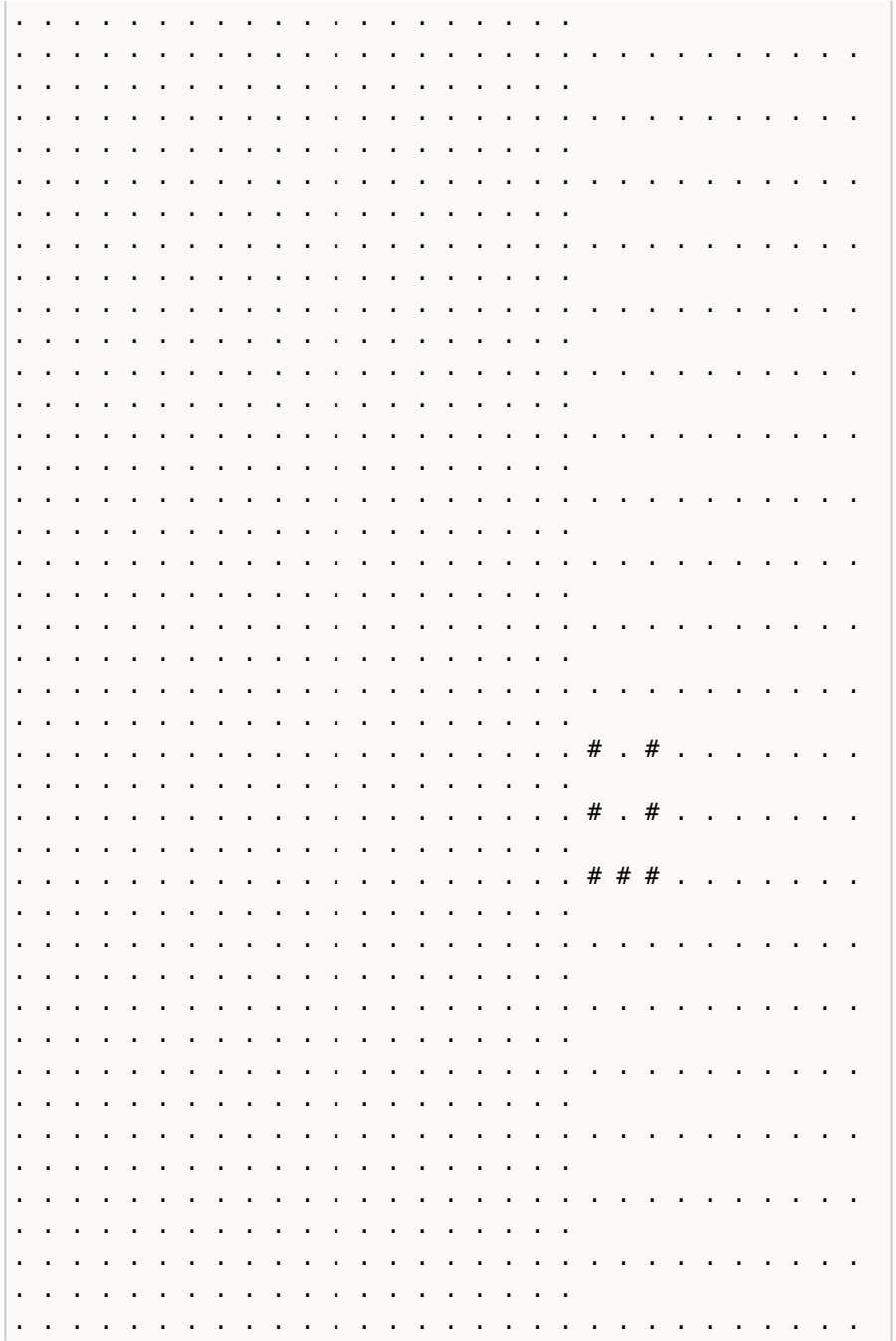
main.cpp

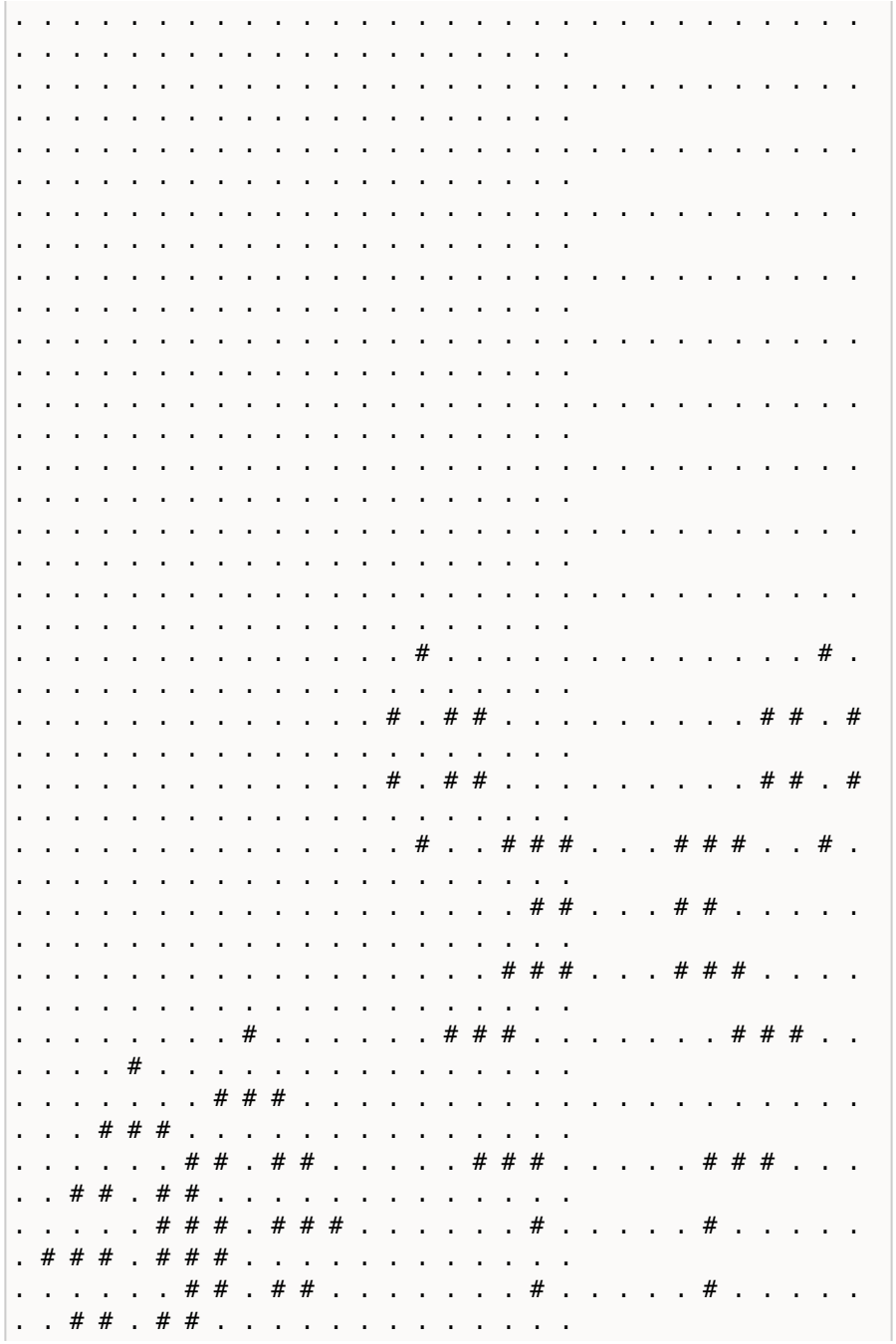
```
#include <iostream>
#include <vector>
#include <algorithm>
#include <random>
using namespace std;

class GameOfLife {
public:
    GameOfLife(unsigned int w, unsigned int h) :
        data(w*h, 0), width(w), height(h)
    {
        at(data, 20, 20) = 1;
        at(data, 20, 21) = 1;
        at(data, 20, 22) = 1;
        at(data, 21, 22) = 1;
        at(data, 22, 22) = 1;
        at(data, 22, 21) = 1;
        at(data, 22, 20) = 1;
    }

    void print() {
        for (unsigned int i {}; i < width*height; ++i) {
            auto const x = i % width;
            auto const y = i / width;
            cout << (at(data, x, y)>0 ? '#' : '.') << (x==
width-1 ? '\n' : ' ');
        }
        cout << endl;
    }

    void next() {
        vector<unsigned char> temp(width * height, 0);
        count_neighbour(data, temp, -1-width);  // left up
        count_neighbour(data, temp,   -width);  // up
```

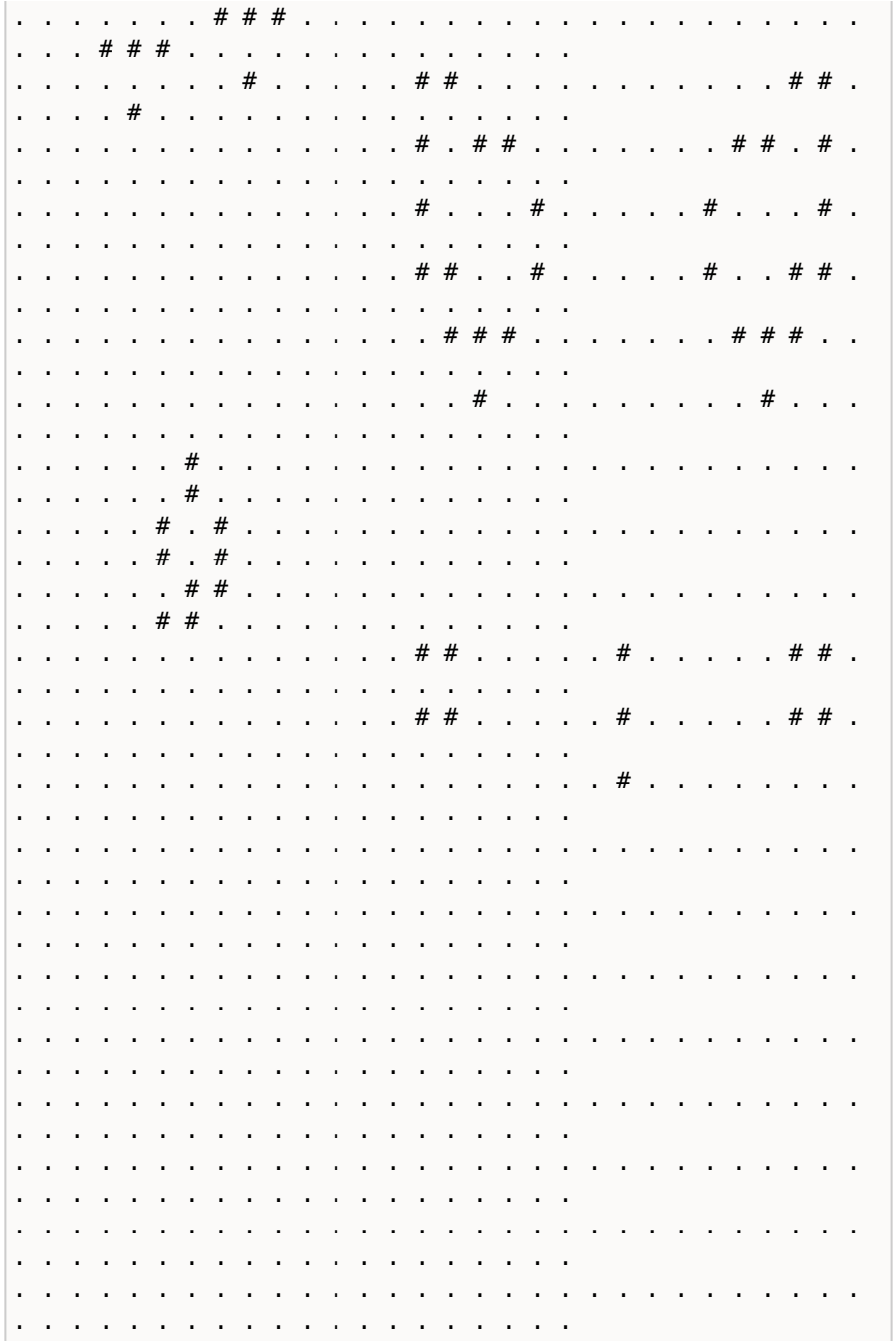
```
        count_neighbour(data, temp,  1-width);  // right up
        count_neighbour(data, temp, -1      );  // left
        count_neighbour(data, temp,  1 );       // right
        count_neighbour(data, temp, -1+width);  // left down
        count_neighbour(data, temp,   +width);  // down
        count_neighbour(data, temp,  1+width);  // right
down
        is_alive(temp, data);
    }

    void random_generate(double p) {
        random_device device;
        mt19937 generator(device());
        std::discrete_distribution<> distribution({1-p, p});
        std::generate(begin(data), end(data), [&](){ return
distribution(generator); });
    }

private:
    unsigned char & at(vector<unsigned char> & v, unsigned
int x, unsigned int y) {
        return v.at(x + width * y);
    }

    void count_neighbour(vector<unsigned char> const& in,
vector<unsigned char> & out, int offset) {
        auto in_it = begin(in);
        auto out_it = begin(out);
        if (offset > 0)
            advance(out_it, offset);
        else
            advance(in_it, -offset);
        for (; in_it != end(in) && out_it != end(out); ++
in_it, ++out_it) {
            *out_it += (*in_it > 0 ? 1 : 0);
        }
    }

    void is_alive(vector<unsigned char> const& in, vector<
unsigned char> & out) {
       transform (begin(in), end(in), begin(out), begin(out),
[](unsigned char i, unsigned char o) {
```

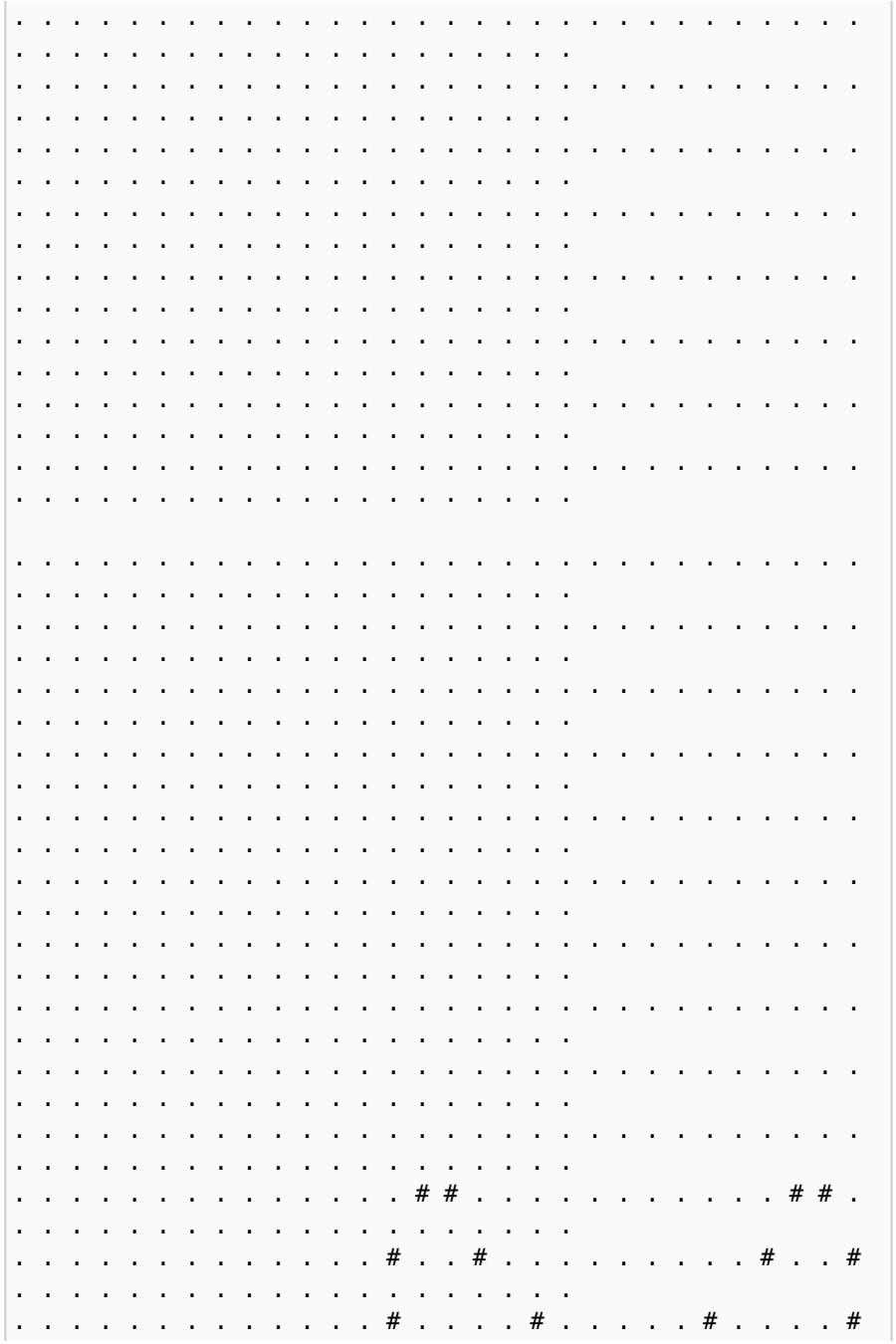
```
            if (o == 0 && i == 3)
                return 1;
            else if ((o == 1 && i == 2) || (o == 1 && i == 3))
                return 1;
            else
                return 0;
        });
    }

private:
    vector<unsigned char> data;
    unsigned int width {};
    unsigned int height {};
};

int main() {
    GameOfLife g(50, 50);
    //g.random_generate(0.1);
    g.print();
    for (int i {}; i<110; ++i) {
        g.next();
        g.print();
    }
}
```

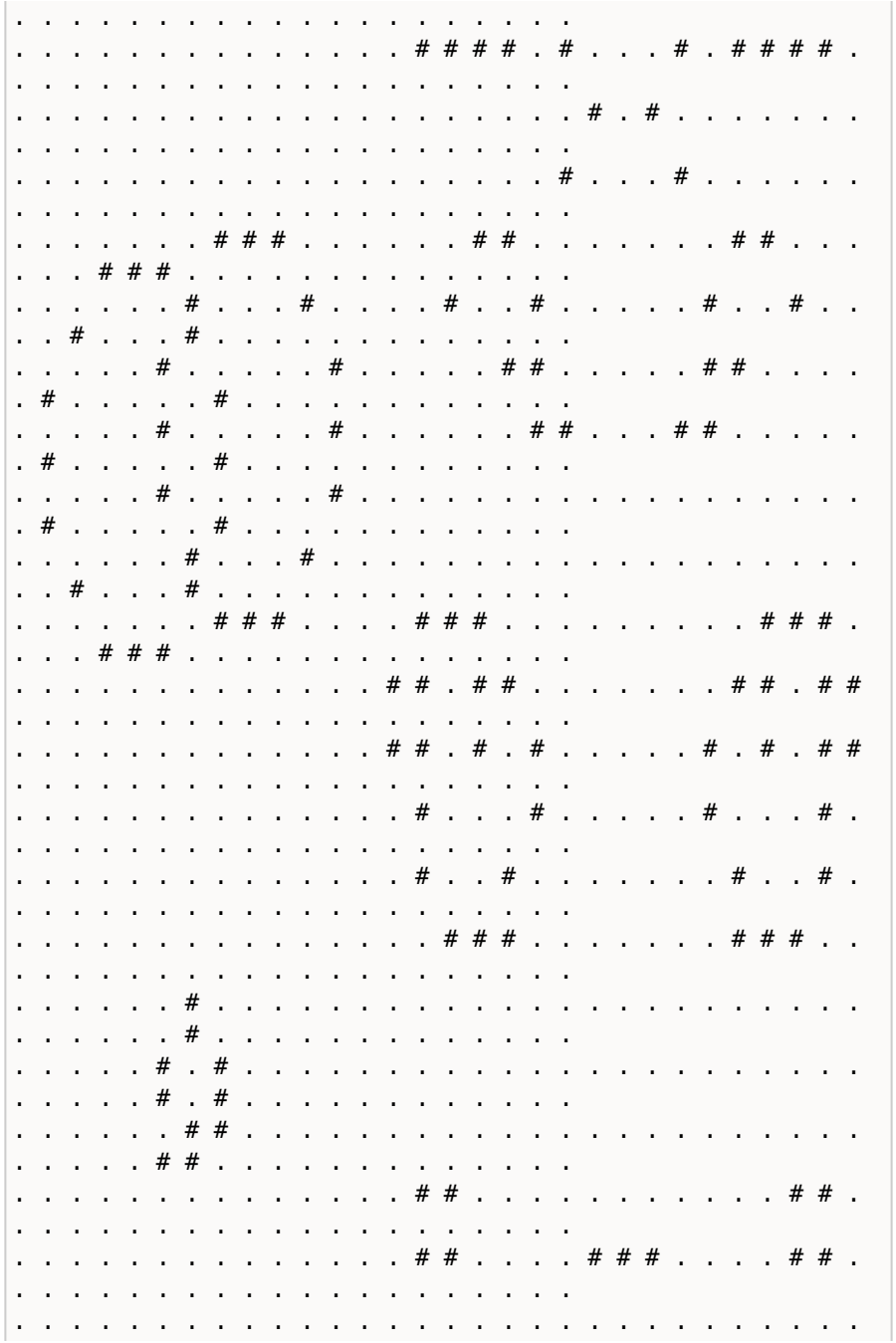
```
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
```

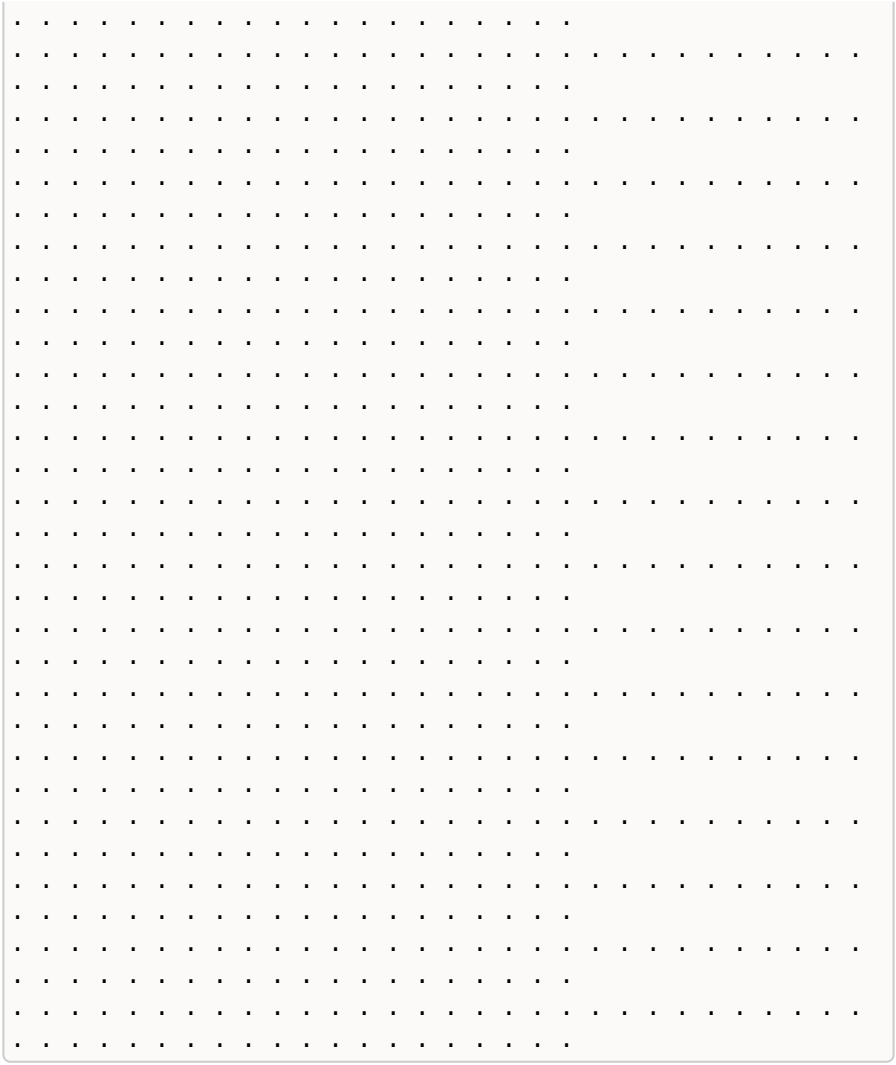












Chapitre précédent	Sommaire principal	Chapitre suivant

Cours, C++