

## Pflicht-Offline-Aufgabe 06-03, INF & WI (MCD: freiwillig):

### Linien zeichnen (rekursiv)

*(geübte C++ Konstrukte: Funktionen, Rekursion,  
Array, for Schleife)*

*Sie brauchen für diese Offline-Aufgabe nur eine einzelne .cpp Datei zu schreiben, in der sowohl die Funktion als auch das Hauptprogramm programmiert sind. Es braucht also keine Headerdatei geschrieben zu werden und Funktionen & Hauptprogramm sollen auch nicht über mehrere Dateien verteilt werden. Dies wäre zwar schöner, würde aber den Arbeitsaufwand für das Hochladen dieser Aufgabe in den Jenkins vergrößern ...*

Schreiben Sie ein C++ Programm, welches den folgenden, hier als „Pseudo-Code“ angegebenen Algorithmus mittels einer Funktion ...

```
void linie(int x1, int y1, int x2, int y2,  
          char canvas[][canvas_size])
```

... realisiert.

**Pseudo-Code:**

```
void linie(int x1, int y1, int x2, int y2,  
          char canvas[][canvas_size])  
{  
    if ( (x1, y1) und (x2, y2) sind benachbart ) {  
        Zeichne die Punkte (x1, y1) und (x2, y2)  
    }  
    else {  
        // Berechne die ganzzahligen Koordinaten des  
        // Punktes in der Mitte zwischen den beiden  
        // Ausgangspunkten:  
        int x_mitte = (x1 + x2)/2;  
        int y_mitte = (y1 + y2)/2;  
  
        // Rekursive Aufrufe:  
        1. Linie vom ersten Punkt bis zur Mitte  
        2. Linie von der Mitte bis zum zweiten Punkt  
    }  
}
```

Wie man berechnet, dass zwei Punkte (x1, y1) und (x2, y2) benachbart sind (d.h. direkt nebeneinander, direkt übereinander oder direkt diagonal zueinander liegen), müssen Sie selbst herausfinden und als C++ Code codieren.

Ein Punkt wird an der Stelle (x, y) gezeichnet, indem das Zeichen `filled_pixel` in das Array `canvas[x][y]` eingetragen wird.

Beachten Sie, dass wegen der Indexzählung im Array ab Null auch die Koordinatenzählung ab Null beginnt, d.h. die linke obere Ecke des Diagramms hat die Koordinate (0, 0).

Benutzen Sie für das gesamte Programm folgendes vorgegebenes Codegerüst, welches Sie nicht verändern dürfen:

```
#include <iostream>
using namespace std;

const char empty_pixel = '.';
const char filled_pixel = '#';

const int canvas_size = 40;

void init_canvas(char canvas[][canvas_size])
{
    for (int x = 0; x < canvas_size; x++)
        for (int y = 0; y < canvas_size; y++)
            canvas[x][y] = empty_pixel;
}

void print_canvas(char canvas[][canvas_size])
{
    for (int y = 0; y < canvas_size; y++) {
        for (int x = 0; x < canvas_size; x++) {
            cout << canvas[x][y];
        }
        cout << endl;
    }
    cout << endl;
}

void linie(int x1, int y1, int x2, int y2, char canvas[][canvas_size] )
{
    // ... Ihr Code hier ...
}
```

```
int main()
{
    char canvas[canvas_size][canvas_size];
    init_canvas(canvas);

    int x1 = 0, y1 = 0, x2 = 0, y2 = 0;
    do {
        cout << "Bitte geben Sie den ersten Punkt ein: ? ";
        cin >> x1 >> y1;
    } while (x1 < 0 || x1 >= canvas_size || y1 < 0 || y1 >= canvas_size);
    do {
        cout << "Bitte geben Sie den zweiten Punkt ein: ? ";
        cin >> x2 >> y2;
    } while (x2 < 0 || x2 >= canvas_size || y2 < 0 || y2 >= canvas_size);

    linie(x1, y1, x2, y2, canvas);

    print_canvas(canvas);

    system("PAUSE");
    return 0;
}
```

Testläufe (Benutzereingaben zur Verdeutlichung unterstrichen):

---

Bitte geben Sie den zweiten Punkt ein: ? 30 35

A 20x20 dot grid with a diagonal sequence of hash symbols (#) starting from the 5th row and 5th column, extending to the 19th row and 19th column.

Drücken Sie eine beliebige Taste . . .

A grid of dots with five hash symbols (#) arranged in a descending staircase pattern from the top-left.

Drücken Sie eine beliebige Taste . . .

A dot grid background with a diagonal sequence of hash symbols (#) starting from the top left and moving towards the bottom right. The sequence consists of 15 hash symbols, each followed by a period, arranged in a descending staircase pattern.

- 6 -