Dokumentation BwInf 2021 - Junioraufgabe 1: „Zum Winde verweht“

Team-ID: 00564

Team-Name: „SRZ info3 Gruppe 1“

Bearbeiter: Karl Jahn

Dresden, der 17. November 2021

Inhalt

1. Lösungsidee

Meine Idee zur Lösung ist recht simpel: Auf der „Karte“ sind alle Häuser verbreitet. Nun gilt es herauszufinden, wie weit jedes dieser Häuser von einem gegebenen Windrad entfernt ist. Da für das Windrad, sowie für die Häuser die Koordinaten gegeben sind, ist dies nicht so schwierig, und über den Satz des Pythagoras’ möglich. Haben wir diese Entfernungen berechnet, können wir überprüfen, welches Gebäude am nächsten am potenziellen Windradstandpunkt gelegen ist. Diese Entfernung ist nur noch mit 10 zu Dividieren und wir erhalten die maximal mögliche Windradhöhe. Nun gilt es nur noch diesen Vorgang für alle Windräder zu Wiederholen.

$P(x) = W(x)$ W = Windrad
$P(y) = H(y)$ H = Haus

$\overline{WP} = W(y) - P(y)$
$\overline{PH} = W(x) - P(x)$

$\overline{WH}^2 = \overline{WP}^2 + \overline{PH}^2$
$\overline{WH} = \sqrt{\overline{WP}^2 + \overline{PH}^2}$

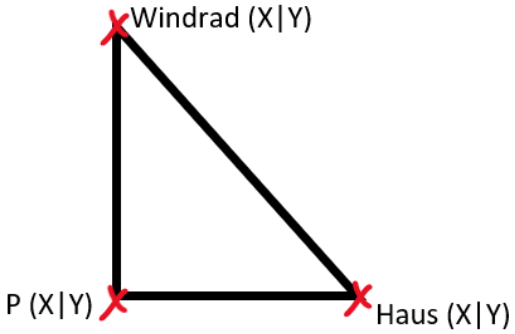

Abb. 1: Hilfsdreieck zur Berechnung der Entfernung

2. Umsetzung

2.1 Algorithmus

Der Algorithmus ist nun ähnlich meiner Idee und in dem Struktogramm unten (Abb. 1) noch einmal nachzuvollziehen.

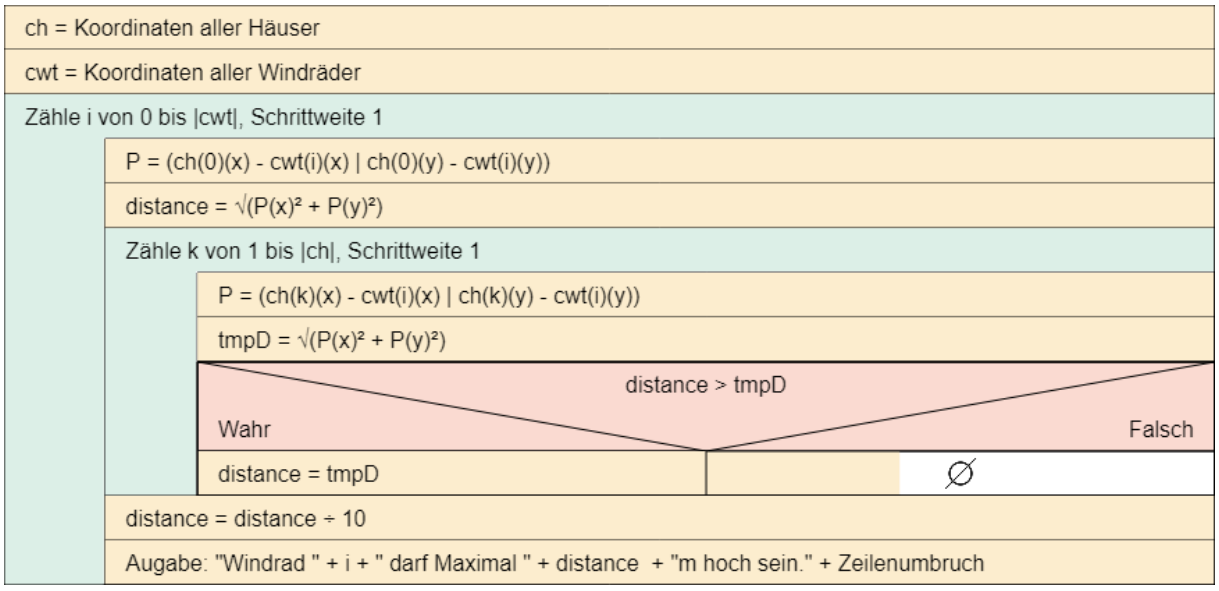

Abb. 2: Algorithmus zur Berechnung der maximalen Höhe der gegebenen Windräder

2.2 Implementation

Zu meiner Implementation, welche ich in C++ (ISO C++ 14-Standard) umsetzte, band ich folgende Standartbibliotheken ein:

iostream	zur ein und Ausgabe in der Konsole
iomanip	Formatierung der Ausgabe
fstream	um die Textdatei lesen
string	Zeichenketten
chrono	zur Zeitmessung

Zusätzlich nutze ich noch die **windows.h**-Bibliothek, d.h. es läuft auch nur auf Windows, um die **system**-Funktion zur Leerung der Konsole verwenden zu können und die Textfarbe und -erscheinung ändern zu können (was die tabellarische Ausgabe vereinfachte).

Die Funktion **readFile** übernimmt den „Dialog“, welcher nach dem Dateipfad fragt. Darauf prüft sie dessen Gültigkeit, liest die Datei, so gültig ein und wandelt deren Inhalt in ein Array um. Dieses Array ist etwas besonders, da ich beim Betrachten der letzten Beispieldatei mit einer höheren Rechenzeit und einem höheren Arbeitsspeicherverbrauch rechnete, speicherte ich die Koordinaten, anstatt in einem zweidimensionalen Array, in einem Eindimensionalen, indem ich zwei **int16_t** (x & y) in einen **uint32_t** via Bitmanipulation/-masking speicherte (siehe Abb. 3, Zeile 4 - 17).

Die Befürchtung mit den höheren Rechen- und Speicheraufwand erwies sich als falsch (63 ms dauerte das Beispiel 4) und mein Trick machte so nur 40 ms aus (was trotzdem einiges ist, aber leider nicht spürbar ist).

Ferner iteriert **readFile** letztendlich durch ebengenanntes Array und ruft die Funktion **findNearest** auf (Definition Abb. 3, Zeile 19 - 25). Diese berechnet nun für jedes Haus die Entfernung zum Windrad.

Das ist eigentlich alles wichtige des Quellcodes, der Rest ist, wenn durch Präprozessor umgeben für die Zeitmessung verantwortlich, oder aber kümmert sich um die Formatierung der Ausgabe, bzw. das Handling des Inputs. Noch anzumerken ist, dass ich die Ausgabe der Windräder mitsamt ihrer Höhe in Zeile 66 - 68 vereinfachte, da es, was die Funktionalität angeht, nichts zur Sache beiträgt.

4. Quellcode

```cpp
int16_t x_diff, y_diff, x, y;
float tmp;

// Abrufen der Abszisse
__forceinline int16_t xValue(uint32_t* coordinate) {
    return (int16_t)(*coordinate >> 16);
}

// Abrufen der Ordinate
__forceinline int16_t yValue(uint32_t* coordinate) {
    return (int16_t)(*coordinate & 0x0000FFFF);
}

// Speichern der Koordinaten
__forceinline uint32_t store(int32_t x, int32_t y) {
    return (uint32_t)x << 16 | ((uint32_t)y & 0x0000FFFF);
}

__forceinline void findNearest(
    uint16_t windTurbine_index, // index des zu bearbeitenden Windrad
    uint32_t* arr,              // das Array mit den Häusern und Windrädern
    size_t* maxNum_houses,
    size_t* house_index,        // der index des nächstgelegen Haus wird gespeichert
    float* distance             // der Abstand dieses Hauses zu dem Windrad
) {

    x_diff = -xValue(&arr[windTurbine_index]),
    y_diff = -yValue(&arr[windTurbine_index]),
    x = xValue(&arr[0]) + x_diff,
    y = yValue(&arr[0]) + y_diff;

    *house_index = 0;
    *distance = std::sqrtf((float)(x * x + y * y));

    for (uint16_t t = 1; t < *maxNum_houses; t++) {
        x = xValue(&arr[t]) + x_diff,
        y = yValue(&arr[t]) + y_diff;

        tmp = std::sqrtf((float)(x * x + y * y));

        if (*distance > tmp) {
            *house_index = t;
            *distance = tmp;
        }
    }
}

void readFile() {

    uint32_t* coordinates;

    size_t
        house_index,    // index des Hauses, welches am nächsten am Windrad ist
        houses,         // Anzahl Häuser
        windTurbines;   // Anzahl Windräder

    float distance;

    // . . . Einlesen der Datei

    for (uint16_t t = 0; t < windTurbines; t++) {
        findNearest(t + houses, coordinates, &houses, &house_index, &distance);

        // Ausgabe (vereinfacht)
        std::cout
            << "Windrad (" << xValue(&coordinates[t]) << "|" << yValue(&coordinates[t])
            << ") darf max " << distance / 10 << "m hoch sein" << std::endl;
    }
}
```

Abb. 3: Quellcode zur Berechnung der maximalen Höhe der Windräder

4. Beispiele

```
BwInf 2021 - Junioraufgabe 1: "Zum Winde verweht"

Team-ID: 00564, Team-Name: "SRZ info3 Gruppe 1"

Bitte geben Sie den Pfad zu der Datei ein: D:\BWINF\Junioraufgabe 1\Beispiel1.txt
Windräder: 3; Häuser: 12
```

Windrad				Haus		
Index	an Position		maximale Höhe	Index	an Position	
1	(1242	-593)	48.52 m	10	(863	-290)
2	(-1223	-1479)	158.98 m	6	(-767	44)
3	(1720	401)	72.41 m	3	(1202	907)

```
benötigte Berechnungszeit: 8 ms, 864 µs
```

Abb. 4: Programmausgabe für „landkreis1.txt“

```
Bitte geben Sie den Pfad zu der Datei ein: D:\BWINF\Junioraufgabe 1\Beispiel2.txt
Windräder: 15; Häuser: 94
```

Windrad				Haus		
Index	an Position		maximale Höhe	Index	an Position	
1	(359	20)	115.16 m	44	(627	1140)
2	(2	-773)	201.25 m	44	(627	1140)
3	(315	-213)	138.85 m	44	(627	1140)
4	(-629	-532)	209.12 m	44	(627	1140)
5	(97	-69)	132.01 m	44	(627	1140)
6	(-392	-418)	186.16 m	44	(627	1140)
7	(87	-384)	161.68 m	44	(627	1140)
8	(-597	612)	133.30 m	44	(627	1140)
9	(-13	-32)	133.54 m	44	(627	1140)
10	(-57	49)	128.77 m	44	(627	1140)
11	(276	292)	91.78 m	44	(627	1140)
12	(156	55)	118.28 m	44	(627	1140)
13	(-423	-93)	161.95 m	44	(627	1140)
14	(202	-219)	142.39 m	44	(627	1140)
15	(-340	-343)	177.04 m	44	(627	1140)

```
benötigte Berechnungszeit: 37 ms, 359 µs
```

Abb. 5: Programmausgabe für „landkreis2.txt“

```
Bitte geben Sie den Pfad zu der Datei ein: D:\BWINF\Junioraufgabe 1\Beispiel3.txt
Windräder: 16; Häuser: 2382
```

Windrad			Haus	
Index	an Position	maximale Höhe	Index	an Position
1	(0 | 0)	451.57 m	2225	(2350 | 3856)
2	(180 | 570)	393.79 m	2225	(2350 | 3856)
3	(360 | 1140)	336.70 m	2225	(2350 | 3856)
4	(540 | 1710)	280.74 m	2225	(2350 | 3856)
5	(360 | -120)	444.62 m	2225	(2350 | 3856)
6	(540 | 450)	385.71 m	2225	(2350 | 3856)
7	(720 | 1020)	327.11 m	2225	(2350 | 3856)
8	(900 | 1590)	269.02 m	2225	(2350 | 3856)
9	(720 | -240)	440.84 m	2225	(2350 | 3856)
10	(900 | 330)	381.25 m	2225	(2350 | 3856)
11	(1080 | 900)	321.73 m	2225	(2350 | 3856)
12	(1260 | 1470)	262.32 m	2225	(2350 | 3856)
13	(1080 | -360)	440.31 m	2225	(2350 | 3856)
14	(1260 | 210)	380.54 m	2225	(2350 | 3856)
15	(1440 | 780)	320.78 m	2225	(2350 | 3856)
16	(1620 | 1350)	261.02 m	2225	(2350 | 3856)

```
benötigte Berechnungszeit: 45 ms, 631 µs
```

Abb. 5: Programmausgabe für „landkreis3.txt“

```
Bitte geben Sie den Pfad zu der Datei ein: D:\BWINF\Junioraufgabe 1\Beispiel4.txt
Windräder: 30; Häuser: 9993
```

Windrad			Haus	
Index	an Position	maximale Höhe	Index	an Position
1	(-4147 | 8575)	0.00 m	1	(-4147 | 8575)
2	(-6453 | 14307)	383.81 m	4309	(-2872 | 12926)
3	(-8370 | 5831)	262.45 m	2213	(-5847 | 6554)
4	(13045 | -5404)	233.99 m	6977	(10804 | -4731)
5	(-8361 | 8131)	296.19 m	9314	(-5404 | 8302)
6	(-6963 | -371)	71.76 m	2052	(-7336 | -984)
7	(9772 | -3239)	181.41 m	6977	(10804 | -4731)
8	(-5102 | -1726)	235.40 m	2052	(-7336 | -984)
9	(13454 | 11822)	343.11 m	5952	(11260 | 9184)
10	(-7427 | 1720)	177.90 m	8720	(-5658 | 1908)
11	(-7816 | 12396)	449.16 m	6420	(-3542 | 11015)
12	(-11095 | 603)	408.03 m	2052	(-7336 | -984)
13	(8314 | 16301)	317.95 m	1280	(7128 | 13351)
14	(15283 | -2961)	221.29 m	9807	(13340 | -1902)
15	(7082 | 18552)	520.12 m	1280	(7128 | 13351)
16	(16743 | 2687)	394.71 m	8517	(13075 | 4145)
17	(17511 | -730)	433.25 m	9807	(13340 | -1902)
18	(-10767 | 12860)	703.83 m	9314	(-5404 | 8302)
19	(1508 | -8030)	168.42 m	1836	(2630 | -6774)
20	(-7767 | 982)	201.27 m	2052	(-7336 | -984)
21	(1277 | -11294)	139.16 m	93	(1980 | -10093)
22	(-8724 | 3575)	348.99 m	8720	(-5658 | 1908)
23	(7033 | -7766)	297.91 m	8812	(8570 | -5214)
24	(2720 | -10910)	110.23 m	93	(1980 | -10093)
25	(20589 | 7265)	813.60 m	8517	(13075 | 4145)
26	(-3214 | 15263)	236.19 m	4309	(-2872 | 12926)
27	(6887 | 17263)	391.94 m	1280	(7128 | 13351)
28	(-3944 | 13584)	125.78 m	4309	(-2872 | 12926)
29	(6576 | 15697)	241.01 m	1280	(7128 | 13351)
30	(-12074 | 5974)	625.40 m	2213	(-5847 | 6554)

```
benötigte Berechnungszeit: 63 ms, 389 µs
Noch eine Datei lesen? [J/N]
```

Abb. 5: Programmausgabe für „landkreis4.txt“

5. Quellen

Abb. 2 wurde mithilfe von https://dditools.inf.tu-dresden.de/ovk/Informatik/Programmierung/Grundlagen/Struktogramme.html erstellt.

Code ist auch auf GitHub verfügbar: https://github.com/Metis-Git/BwInf-2021/blob/main/Junioraufgabe1/src/Junioraufgabe1.cpp