

Lösung zu Übungsblatt 2

Aufgabe 1:

Was in dieser Aufgabe verlangt wird, ist, dass man errechnet, welcher Algorithmus jeweils für die angegebenen n am besten sind (und wir werden sehen, dass man nicht immer generell sagen kann, dass ein Algorithmus besser ist als der andere, sondern dass das tatsächlich von der Größe n abhängt).

Was wir dazu machen ist folgendes: Wir setzen für n jeweils den gegebenen Wert ein und schauen, was sich dann ergibt:

Für $n = 50$:

$$f1(50) = 150 * 50 = 7500$$

$$f2(50) = 2 * (50)^2 = \mathbf{5000}$$

$$f3(50) = 50^3 / 10 = 125000 / 10 = 12500$$

In diesem Fall ist also $O(n^2)$, also $f2(n)$ der beste Algorithmus.

Betrachten wir den zweiten Fall ($n=1000$):

$$f1(1000) = 150 * 1000 = 150.000$$

$$f2(1000) = 2 * (1000)^2 = 2.000.000$$

$$f3(1000) = 1000^3 / 10 = 100.000.000$$

In diesem Fall ist Algorithmus 1 ($f1(n) = O(n)$) der beste.

In beiden Fällen (also sowohl bei $n = 50$ als auch für $n = 1000$) ist $f3(n) (= O(n^3))$ die schlechteste Lösung.

Damit wird deutlich:

$O(n^2)$ muss nicht unbedingt schlechter sein als $O(n)$ → Entscheidend ist die Größe von n !

Aufgabe 2:

Zu erst einmal der Quelltext:

```
#include <iostream>
```

```
#include <conio.h>
```

```
#include <cmath>
```

```
using namespace std;
```

```
float hoehe_berechnen(float distanz, float winkel)
```

```
{
```

```
    float erg;
```

```
    erg = distanz * tan(winkel);
```

```
    return erg;
```

```
}
```

```

main()
{
    float distanz, winkel, hoehe;
    cout << "Entfernung: ";
    cin >> distanz;

    cout << endl << "Winkel: ";
    cin >> winkel;

    winkel = winkel * 2 * 3.14159/360;

    hoehe = hoehe_berechnen(distanz, winkel);

    cout << "Hoehe des Bungalows: " << hoehe << endl;

    getch();
}

```

Gehen wir das Ganze jetzt mal Schritt für Schritt durch:
Wir beginnen natürlich wieder in der main-Funktion:

In der Aufgabe ist die Rede von Höhe, Winkel und Distanz. Das sind drei Variablen. Die legen wir erstmal an!

```

float distanz, winkel, hoehe;

```

Man hätte die drei Variablen auch einzeln „deklarieren“ können, aber man kann sie auch, wie hier gezeigt, in einem Schritt, durch Kommas getrennt, deklarieren.

Schreiben wir das Programm am besten gleich so, dass der Benutzer einen Wert für die Entfernung selbst eingeben kann.

Dann müssen wir ja mit dem Benutzer kommunizieren und ihm sagen, dass er die Entfernung eingeben soll.

Also:

```

cout << "Entfernung: ";

```

Dann soll er die Entfernung auch eingeben können – logisch.

```

cin >> distanz;

```

Die Eingabe, die der Benutzer jetzt tätigt, wird also in der Variable **distanz** gespeichert.

Genau so verfahren wir nun mit dem Winkel:

```

    cout << endl << "Winkel: ";
// ( Das endl habe ich aus rein ästhetischen Gründen mit eingebracht...)
    cin >> winkel;

```

Nun hieß es in der Aufgabe, dass der Winkel für die tan-Funktion aus der cmath-Bibliothek die Winkelangabe in rad braucht.

Mit 25 wird der Benutzer ja den Winkel in Gradmaß angeben.

Hier muss also noch eine Umrechnung passieren.

Dazu folgende Überlegung:

Die komplette Umdrehung wären 360 Grad.

Im Bogenmaß wäre sie 2π .

Nun ist 25 Grad im Verhältnis zu der kompletten Umdrehung, also zu 360 Grad das gleiche wie der uns unbekannte Wert im Bogenmaß im Verhältnis zu der kompletten Umdrehung 2π .

Also ergibt sich folgende Gleichung:

$$25/360 = x/2\pi.$$

Das Ganze müssen wir jetzt nach x umrechnen:

Somit ergibt sich:

$$x = 25 * 2\pi / 360$$

Das wäre dann der Wert des Winkels im Bogenmaß.

Dieser Winkel soll ja in der Variable **Winkel** gespeichert werden (wo bisher 25.0 gespeichert ist - wenn der Benutzer 25 eingegeben hat, logischer Weise).

Also:

$$\text{winkel} = \text{winkel} * 2 * 3.14159 / 360;$$

Die Berechnung der Höhe machen wir jetzt mal in einer Funktion.

(Wenn Du die noch nicht selbst geschrieben hast – vielleicht kannst Du es ja jetzt – versuch es also erstmal und ließ dann weiter).

In die Variable **hoehe** soll jetzt die berechnete Höhe gespeichert werden.

In der Klausur hieße es jetzt wie folgt:

Schreibe eine Funktion **float** hoehe_berechnen(**float** distanz, **float** winkel). Mit den übergebenen Parametern soll sie die Höhe berechnen und das Ergebnis zurückgeben.

Der Aufruf für diese Funktion wäre:

$$\text{hoehe} = \text{hoehe_berechnen}(\text{distanz}, \text{winkel});$$

Nehmen wir jetzt mal an, die Höhe wurde jetzt berechnet (wir schauen uns die Funktion selbst gleich noch genauer an) – dann müssen wir ja nur noch das Ergebnis ausgeben, also:

$$\text{cout} \ll \text{"Hoehe des Bungalows: "} \ll \text{hoehe} \ll \text{endl};$$

und, nicht vergessen:

```
    getch();  
}
```

So, jetzt aber zur Funktion **float** hoehe_berechnen(**float** distanz, **float** winkel):

Das einzige, was jetzt noch passieren muss, ist die Berechnung der Höhe nach der Formel:

Höhe = distanz * tan(alpha).

Also gehen wir wie folgt vor:

Wir schaffen uns eine Variable, in der wir das Ergebnis speichern, nennen wir sie mal erg (sie muss natürlich vom Typ float sein, denn sie soll ja auch Kommazahlen annehmen können).

```
float erg;
```

in dieser Variablen speichern wir nun das, was bei der Berechnung rauskommt, also:

```
erg = distanz * tan(winkel);
```

Anschließend geben wir genau dieses Ergebnis zurück:

```
return erg;
```

Alternativ hätte man das ganze auch komplett in einer Zeile schreiben können:

```
return (distanz*tan(winkel));
```

Somit hätten wir aber lediglich den Speicherplatz für erg gespart – aber nun...