



1. Deklaration des Struktur-Datentyps

Strukturen fassen mehrere primitive oder komplexe Variablen zu einer logischen Einheit zusammen. Die Variablen dürfen dabei unterschiedliche Datentypen besitzen. Die einzelnen Variablen der Struktur (mit ihrem Datentyp) werden als **Komponenten** (engl. members) bezeichnet.

Die Größe einer Variable vom Typ **struct** StrukturName kann mit **sizeof** ermittelt werden.

```
⑧ sizeof(struct StrukturName);
```

```
① :  
② struct StrukturName  
③ {  
④     Datentyp NameKomponente_1;  
⑤     Datentyp NameKomponente_2;  
⑥     :  
⑦     Datentyp NameKomponente_N;  
};  
:  
:  
Deklaration.cpp
```

Dabei kann die Größe eines **struct**-Typs mehr sein als die Summe der einzelnen Komponenten. Der Compiler richtet die einzelnen Komponenten so im Speicher aus, dass ein schneller Zugriff möglich ist.
Quelle : "de.wikibooks.org/wiki/C-Programmierung"

2. Definition von Strukturvariablen

Nach der Deklaration des Struktur-Datentyps können (eine oder mehrere) Variablen des Typs angelegt werden.

```
⑨ struct StrukturName VariablenName ;  
⑩ struct StrukturName Variable_1 , Variable_2 ... Variable_M;
```

Durch weitere Syntaxvarianten ist es auch möglich Struktur-Datentyp und Variable(en) mit einer Anweisung zu deklarieren. Bei dieser Variante ist auch die namenlose Typendeklaration realisierbar (wird hier nicht betrachtet).

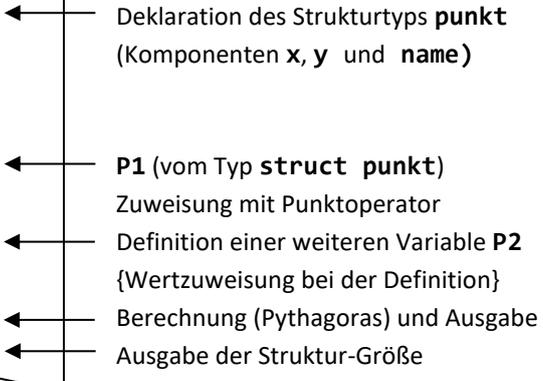
3. Wertzuweisung mit Punktoperator

Um den einzelnen Struktur-Komponenten einer Strukturvariablen Werte zu zuweisen, wird mittels des Punktoperators zugegriffen.

```
⑪ VariablenName.NameKomponente_1 = Wert_1;  
⑫ VariablenName.NameKomponente_2 = Wert_2;  
⑬ :  
⑭ VariablenName.NameKomponente_N = Wert_N;
```

4. Zusammenfassendes Beispiel einer Struktur mit Strukturvariablen und Wertzuweisung

```
① # include <iostream>  
② # include <cmath>  
③ using namespace std;  
④ int main() {  
⑤     struct punkt {  
⑥         int x;  
⑦         int y;  
⑧         string name;  
⑨     } P1;  
⑩     P1.x = 2 ; P1.y=1 ; P1.name = "A";  
⑪     struct punkt P2 = {3,4,"E"};  
⑫     cout <<"von"<<P1.name<<"bis"<<P2.name ;  
⑬     cout << hypot((P2.x-P1.x),(P2.y-P1.y));  
⑭     cout << sizeof(struct punkt);  
⑮ }  
PunktAbstand.cpp
```



Konsole-Ausgabe:
Abstand P1-P2 = 3.16
Groesse = 20

