

Vorlesung 11

- Struct (Strukturen)
 - Definition
 - Verwendung
- Zeit im Computer
 - Zeitmessung

Strukturen: ein neuer Begriff

Viele Programmieraufgaben lassen sich mit den so genannten **Strukturen**, einem speziellen Datenkonstrukt, leichter lösen.

Def1: Eine Struktur ist ein vom Programmierer definierter Datentyp.

Strukturen: ein neuer Begriff

Def2: Eine Struktur ist eine Sammlung von einer oder mehreren Variablen, die zum Zweck der leichteren Manipulation unter einem Namen zusammengefasst werden.

Anmerkungen

- i) Die Variablen in einer Struktur können im Gegensatz zu denen in einem Array unterschiedlichen Datentypen angehören.
- ii) Eine Struktur kann jeden beliebigen C++-Datentyp enthalten.

Strukturen: die Grundbegriffe

- Struktur **definieren** und **benennen**.
- Variablen** mittels Struktur deklarieren.
- Zugriff** auf Daten (**Komponenten**) in Strukturen.

Struktur: Aufgabe

Aufgabe:

Es soll ein Programm geschrieben werden.

Das Programm soll eine Struktur

Name *Person*

mit drei Komponenten

Alter,

Größe

und *Gewicht*

definieren.

Struktur definieren

Selbst definierte Datentyp
-Struct

Lösungsweg

Name der Struktur

```
struct Person
```

```
{
```

```
    int Alter;
```

```
    float Groesse;
```

```
    float Gewicht;
```

```
};
```

Komponenten

Struktur definieren

Um eine Struktur zu definieren, braucht man:

- das Schlüsselwort **struct**.
- Den **Name** der Struktur.
- Die geschweiften Klammern.
- Die **Komponenten** der Struktur.
- Das Semikolon am Ende

Struktur benutzen um Variablen zu vereinbaren

Selbstdefinierte Typen - struct

Schablone definieren

```
struct Person
{
    int Alter;
    float Groesse;
    float Gewicht;
};
```

Schablone nutzen um Variablen zu vereinbaren

```
person student, dozent, Arzt;
```


Auf Strukturen-Komponenten zugreifen:

Um Strukturen sinnvoll nutzen zu können, sollte man ihren Komponenten Werte zuweisen.

Zu diesem Zweck kennt C/C++
den **Punkt-Operator: .**



Auf Strukturen-Komponenten zugreifen:

DEFINITION der struct Person

```
struct Person
{
    int Alter;
    float Groesse;
    float Gewicht;
};
```

DEKLARATION der Variable student vom Datentyp Person

```
person student;
```

```
student.Alter=18;
student.Gewicht=50.70;
student.Groesse=1.80;
```

The diagram illustrates the relationship between the struct definition, the variable declaration, and the field access code. A dashed arrow points from the struct definition box to the access code box, and another dashed arrow points from the variable declaration box to the same access code box, indicating that both are necessary for the code to compile and run correctly.

```
[ _ _ _ _ _ ]
[ personStruktur.cpp ]
```

Komplexere Strukturen:

Arrays in Struktur

```
struct Person
{
    char name[31];
    char vname[21];
    char gebdatum[9];
    char geschlecht:
    int MatN;
}student;
```

Die Struktur kann auch
Arrays beinhalten

Erweiterung der Komponenten

- Einer *struct* kann man Komponenten hinzufügen, ohne den Programmcode wesentlich zu verändern.

Erweiterung der Komponenten

Name der Struktur

```
struct Person
```

```
{
```

```
    int Alter;
```

```
    float Groesse;
```

```
    float Gewicht;
```

```
    char Name[25];
```

```
    bool istStudiumFertig;
```

```
    char StudiumAbschluss[25];
```

```
};
```



**Alte
Komponenten**



**neue
Komponenten**

| *personStrukturErweitert.cpp* |

Zeit im Rechner

Die einfachsten Funktionen für die Zeitverwaltung, -abruf sind in der Bibliothek

ctime in C++ (bzw. time.h in C)

2 Aufgaben sind uns erstens möglich

- Zeitabmessung eines Zeitintervalls
- Ermittlung des Datums

Zeit im Rechner: Zeitmessung

- Die Funktion `clock()` gibt die Anzahl von CPU Ticks zurück
- Der Datentyp für die Uhrticks ist `clock_t`
- Das Verhältnis zwischen Sekunden und Uhrticks ist `CLOCKS_PER_SEC`

Uhrticks auffordern

- Ich muss eine Variable `clock_t` und den Aufruf von `clock()` da speichern.

```
clock_t zeit1, zeit2;  
float sek;  
zeit1 = clock();
```

...Ein bestimmtes Prozess

```
zeit2 =clock();  
sek = (float) (zeit2 - zeit1) / CLOCK_PER_SEC
```

- Anmerkung „(float)“ macht eine Division mit Kommazahl.
Ohne „(float)“ ist das Ergebnis eine ganze Zahl.

| Quiz.cpp |