

UNIVERSITÄT DORTMUND

Wintersemester 2007/08

**Einführung in die Informatik für
Naturwissenschaftler und Ingenieure**
(alias Einführung in die Programmierung)
(Vorlesung)

Prof. Dr. Günter Rudolph
Fachbereich Informatik
Lehrstuhl für Algorithm Engineering

Kapitel 2: Darstellung von Information

Inhalt

- Zusammengesetzte Datentypen
 - Feld (array)
 - Verbund (struct)
 - Aufzählung (enum)

Rudolph: EINI (WS 2007/08) • Kap. 2: Darstellung von Information 2

Darstellung von Information

Zusammengesetzte Datentypen

- **Array (Feld)**
 - Einführendes Beispiel:
Temperaturen von gestern stündlich speichern

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
8.4	8.3	8.0	7.4	7.2	7.0	7.0	7.5	8.0	8.8	9.8	11.1	13.4	13.6	13.7	13.6	12.4	12.0	10.1	9.6	9.0	8.9	8.7	8.5

- Möglicher Ansatz:


```
float x00, x01, x02, x03, x04, x05, x06, x07,
      x08, x09, x10, x11, x12, x13, x14, x15,
      x16, x17, x18, x19, x20, x21, x22, x23;
```
- Besser:

Unter einem Namen zusammenfassen und zur Unterscheidung der Werte einen Index verwenden.

Rudolph: EINI (WS 2007/08) • Kap. 2: Darstellung von Information 3

Darstellung von Information

Array

Datendefinition: `float x[24];`

Anzahl bereitzustellender Speicherplätze

Gemeinsamer Datentyp

Gemeinsamer Bezeichner

- Zugriff auf das Feldelement: `x[12];`

Achtung:

- Der Index beginnt immer bei 0!
- `x[12]` greift also auf das 13. Feldelement zu!
- Der maximale Index wäre hier also 23.
- Was passiert bei Verwendung von `x[24]`? ⇒ ABSTURZ!

Fataler Fehler!

Rudolph: EINI (WS 2007/08) • Kap. 2: Darstellung von Information 4

Eindimensionales Array

- Ein **Array** ist eine Aneinanderreihung von **identischen** Datentypen
 - mit einer **vorgegebenen Anzahl** und
 - unter einem **gemeinsamen Bezeichner**.
- Der Zugriff auf einzelne Elemente erfolgt über einen **Index**
 - der **immer bei 0** beginnt und
 - dessen **maximaler Wert** genau **Anzahl - 1** ist.
- (Fast) alle Datentypen können verwendet werden.

Eindimensionales Array: Beispiele

- `unsigned int Lotto[6];`
- `double Monatsmittel[12];`
- `char Vorname[20];`
- `bool Doppelgarage_belegt[2];`

- **Datendefinition**
Datentyp Bezeichner[Anzahl];

Eindimensionales Array: Initialisierung

- `unsigned int Lotto[6] = { 27, 10, 20, 5, 14, 15 };`
- `unsigned int Lotto[] = { 27, 10 };` ← **Compiler ermittelt erforderliche Anzahl**
- `unsigned int Lotto[6] = { 27, 10 };`
ist identisch zu
`unsigned int Lotto[6] = { 27, 10, 0, 0, 0, 0 };`
- `unsigned int Lotto[6] = { 0 };`
ist identisch zu
`unsigned int Lotto[6] = { 0, 0, 0, 0, 0, 0 };`

Eindimensionales Array: Verwendung

```
float Temp[12] = { 2.3, 4.6, 8.9, 12.8 };
float x, y, z = 1.2;
Temp[4] = z;
x = Temp[0] * 0.25;
y = Temp[1] + 2.3 * Temp[2];
int i = 2, j = 3, k = 4, m = 11;
z = ( Temp[i] + Temp[j] + Temp[k] ) / 3.0;
Temp[m] = z + Temp[k - i];
```

Eindimensionales Array: Verwendung

```
float Temp[12] = { 2.3, 4.6, 8.9, 12.8 };
float TempNeu[12];
TempNeu = Temp;
```



Merken!

- Ein Array kann nicht als Ganzes einem anderen Array zugewiesen werden!
- Eine Zuweisung muss immer elementweise verfolgen!

Zwei- und mehrdimensionales Array

• Einführendes Beispiel

- Pro Tag drei Temperaturmessungen: morgens, mittags, abends
- Werte für eine Woche (7 Tage) ablegen

⇒

8.0	20.3	14.2
7.8	18.3	12.2
5.3	12.3	8.8
5.8	13.7	7.5
8.0	19.8	10.2
9.3	21.3	11.1
7.4	17.3	9.9

Tabelle
oder
Matrix
der Temperaturen

Zwei- und mehrdimensionales Array

• Einführendes Beispiel

```
float tag0[3], tag1[3], tag2[3] usw. bis tag6[3];
```

	0	1	2
tag0	8.0	20.3	14.2
tag1	7.8	18.3	12.2
tag2	5.3	12.3	8.8
tag3	5.8	13.7	7.5
tag4	8.0	19.8	10.2
tag5	9.3	21.3	11.1
tag6	7.4	17.3	9.9

Zwei- und mehrdimensionales Array

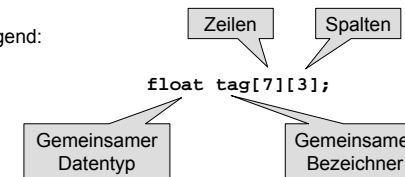
• Einführendes Beispiel

- Statt

```
float tag0[3], tag1[3], tag2[3] usw. bis tag6[3];
```

bräuhcte man ein Array von Arrays vom Typ float!

- Nahe liegend:



Zwei- und mehrdimensionales Array

• Einführendes Beispiel

⇒ Spaltenindex

	0	1	2	
⇒ Zeilenindex	0	8.0	20.3	14.2
	1	7.8	18.3	12.2
	2	5.3	12.3	8.8
	3	5.8	13.7	7.5
	4	8.0	19.8	10.2
	5	9.3	21.3	11.1
	6	7.4	17.3	9.9

tag[0][2] hat Wert 14.2
 tag[2][0] hat Wert 5.3
 tag[4][2] hat Wert 10.2
 tag[2][4] ist ungültig!

Zwei- und mehrdimensionales Array

• Initialisierung

```
float tag[7][3] = {
    { 8.0, 20.3, 14.2 },
    { 7.8, 18.3, 12.2 },
    { 5.3, 12.3, 8.8 },
    { 5.8, 13.7, 7.5 },
    { 8.0, 19.8, 10.2 },
    { 9.3, 21.3, 11.1 },
    { 7.4, 17.3, 9.9 }
};
```

oder

```
float tag[][3] = {
    { 8.0, 20.3, 14.2 },
    { 7.8, 18.3, 12.2 },
    { 5.3, 12.3, 8.8 },
    { 5.8, 13.7, 7.5 },
    { 8.0, 19.8, 10.2 },
    { 9.3, 21.3, 11.1 },
    { 7.4, 17.3, 9.9 }
};
```

Zwei- und mehrdimensionales Array

• Datendefinition bei ansteigender Dimension

1. int feld[n];
2. int feld[m][n];
3. int feld[k][m][n];
4. usw.

Zusammengesetzte Datentypen

• Zeichenkette

- ... ist eine Aneinanderreihung von Zeichen
- ⇒ also ein Array/Feld von Zeichen

Datendefinition: char wohnort[40];

Initialisierung:

```
char wohnort[40] = {'D','o','r','t','m','u','n','d','\0'};
char wohnort[40] = "Dortmund";
char wohnort[] = "Dortmund";
```

kennzeichnet Ende der Zeichenkette

riskant!

sicher: Compiler zählt!

• Zeichenkette

- Das Ende wird durch das ASCII Steuerzeichen NUL (mit Code 0) gekennzeichnet!
- => Bei der Datendefinition muss also **immer ein Zeichen mehr** angefordert werden als zur Speicherung der Daten benötigt wird!

Falsch ist: `char wort[3] = "abc";`

- Zuweisung einer Zeichenkette an eine andere nicht zulässig (weil array von `char`)

Falsch ist: `char wort[4]; wort[4] = "abc";`
oder: `wort[] = "abc";`

- Zuweisung muss immer **elementweise** erfolgen!

Beispiel: `char wort[4] = "abc"; wort[0] = 'z';`

Zusammengesetzte Datentypen

• Datenverbund (Struktur)

- Einführendes Beispiel:
Zu speichern sei Namen und Matrikelnummer von Studierenden und ob Vordiplom bestanden ist

■ Möglicher Ansatz:

Drei verschiedene Datentypen (`char[]`, `unsigned int`, `bool`)
=> in Array lässt sich nur ein gemeinsamer Datentyp speichern
=> alles als Zeichenketten, z.B. `char stud[3][40];`

■ Besser:

Zusammen gehörende Daten unter einem Namen zusammenfassen aber die „natürlichen“ Datentypen verwenden!

Zusammengesetzte Datentypen

• Datenverbund (Struktur)

- Wir definieren uns unseren eigenen Datentyp!
- Wir müssen die Struktur / den Bauplan definieren!
- Wir müssen einen Namen für den Datentyp vergeben!

```
struct UnserDatenTyp
{
    char name[40];
    unsigned int matrikel;
    bool vordiplom;
};
```

← Name des Datentyps

Bauplan / Struktur

Zusammengesetzte Datentypen

• Datenverbund (Struktur)

- Zuerst das Schlüsselwort: `struct`
- Dann folgt der gewählte Name (engl. *tag*).
- In geschweiften Klammern `{}` steht der Bauplan. Am Ende ein Semikolon ;

```
struct UnserDatenTyp
{
    char name[40];
    unsigned int matrikel;
    bool vordiplom;
};
```

← Name des Datentyps

Bauplan / Struktur

Datenverbund (Struktur)

- **Achtung:**
Soeben wurde ein Datentyp definiert.
Es wurde noch **kein Speicherplatz** reserviert!
- Datendefinition:
`UnserDatentyp student, stud[50000];`
- Initialisierung:
`UnserDatentyp student = { "Hugo Hase", 44221, true };`
- Zugriff mit „Punktoperator“:
`unsigned int mnr = student.matrikel;`
`cout << student.name << " " << mnr << endl;`

Reihenfolge beachten!

Datenverbund (Struktur)

- Im Bauplan kann wieder jeder Datentyp vorkommen!
- Also auch wieder Datenverbunde (`struct`)!
- Beispiel:
`struct UniStud {
char ort[40];
unsigned int plz;
UnserDatentyp daten;
};`

`UniStud studX = {
"Dortmund", 44221, { "Jane Doe", 241398, true }
};`

`unsigned int mnr = studX.daten.matrikel;`

Datenverbund (Struktur)

- Zuweisungen:
`UnserDatentyp stud[50000];
UnserDatentyp student = { "Hugo Hase", 44221, true };
stud[500] = student;
student = stud[501];`
- Ganze Datensätze können strukturidentischen Variablen zugewiesen werden.
Komponentenweises Zuweisen nicht nötig!
- **Achtung:**
Anderer Name (tag) ⇒ Anderer Datentyp!
Gilt selbst bei identischen Bauplänen!
`struct S1 { int x; float y; };
struct S2 { int x; float y; };
S1 v1, vx; v1 = vx;
S2 v2; v2 = vx;`

Fehler!

Zusammengesetzte Datentypen

- **Aufzähltyp (enum)**
 - Umwelt beschreiben durch Begriffe statt durch Ziffern.
 - Farben: rot, blau, grün, orange, gelb, schwarz, ...
 - Spielkarten: Kreuz, Pik, Herz, Karo.
 - Internet-Domains: de, uk, fr, ch, fi, ru, ...
1. Schlüsselwort `enum` (Enumeration, Aufzählung)
 2. Name der Aufzählung
 3. In geschweiften Klammern die Elementnamen.
- ```
enum Kartentyp { kreuz, pik, herz, karo };
```

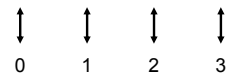


### Zusammengesetzte Datentypen

- **Aufzähltyp (enum)**

- Was passiert im Rechner?
- Interne Zuordnung von Zahlen (ein Code)

```
enum KartenTyp { kreuz, pik, herz, karo };
```



- Zuordnung der Zahlen durch Programmierer kontrollierbar:

```
enum KartenTyp { kreuz=1, pik=2, herz=4, karo=8 };
```

- Initialisierung: `KartenTyp Spielfarbe = kreuz;`
- Aber: `cout << Spielfarbe << endl;`  
Ausgabe ist Zahl!