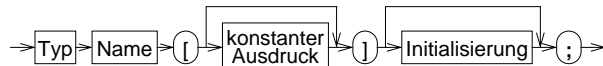


G Felder und Strukturen

G.1 Eindimensionale Felder

- eine Reihe von Daten desselben Typs kann zu einem **Feld** zusammengefaßt werden
- bei der Definition wird die Anzahl der Feldelemente angegeben, die Anzahl ist konstant!
- der Zugriff auf die Elemente erfolgt durch **Indizierung**, beginnend bei Null
- Definition eines Feldes



- Beispiele:

```
int x[5];  
double f[20];
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.1

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.2 ... Initialisierung eines Feldes (2)

- Felder der Speicherklasse *automatic* (lokale Felder) können in altem K&R-C nicht initialisiert werden
ANSI-C erlaubt die Initialisierung von Feldern
- Felder des Typs **char** können auch durch String-Konstanten initialisiert werden

```
char name1[5] = "Otto";  
char name2[] = "Otto";
```

C-Ing

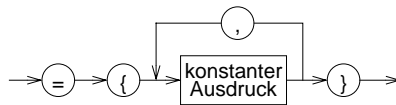
Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.3

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.2 Initialisierung eines Feldes



- Ein Feld kann durch eine Liste von konstanten Ausdrücken, die durch Komma getrennt sind, initialisiert werden

```
int prim[4] = {2, 3, 5, 7};  
char name[5] = {'o', 't', 't', 'o', '\0'};
```

- wird die explizite Felddimensionierung weggelassen, so bestimmt die Zahl der Initialisierungskonstanten die Feldgröße

```
int prim[] = {2, 3, 5, 7};  
char name[] = {'o', 't', 't', 'o', '\0'};
```

- werden zu wenig Initialisierungskonstanten angegeben, so werden die restlichen Elemente mit 0 initialisiert

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

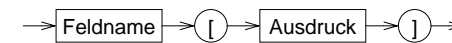
G-Felder.doc 1998-06-24 10.04

G.2

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.3 Zugriffe auf Feldelemente

- Indizierung:



wobei: $0 \leq \text{Wert}(\text{Ausdruck}) < \text{Feldgröße}$

- Beispiele:

```
prim[0] == '2'  
prim[1] == '3'  
name[1] == 't'  
name[4] == '\0'
```

- Beispiel Vektoraddition:

```
float v1[4], v2[4], sum[4];  
int i;  
...  
for ( i=0; i < 4; i++ )  
    sum[i] = v1[i] + v2[i];  
for ( i=0; i < 4; i++ )  
    printf("sum[%d] = %f\n", i, sum[i]);
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

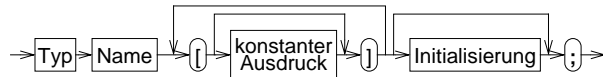
G.4

Reproduktion jeder Art oder Verwendung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.4 Mehrdimensionale Felder

- neben eindimensionalen Felder kann man auch mehrdimensionale Felder vereinbaren
 - ◆ ihre Bedeutung in C ist gering

- Definition eines mehrdimensionalen Feldes



- Beispiel:

```
int matrix[4][4];
```

2 Initialisierung eines mehrdimensionalen Feldes

G.4 Mehrdimensionale Felder

- ein mehrdimensionales Feld kann - wie ein eindimensionales Feld - durch eine Liste von konstanten Werten, die durch Komma getrennt sind, initialisiert werden
- wird die explizite Felddimensionierung weggelassen, so bestimmt die Zahl der Initialisierungskonstanten die Größe des Feldes

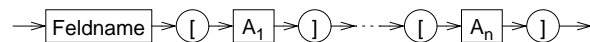
- Beispiel:

```
int feld[3][4] = {  
  { 1, 3, 5, 7}, /* feld[0][0-3] */  
  { 2, 4, 6 } /* feld[1][0-2] */  
};  
feld[1][3] und feld[2][0-3] werden in dem Beispiel nicht initialisiert!
```

1 Zugriffe auf Feldelemente

G.4 Mehrdimensionale Felder

- Indizierung:



wobei: $0 \leq A_i < \text{Größe der Dimension } i \text{ des Feldes}$
 $n = \text{Anzahl der Dimensionen des Feldes}$

- Beispiel:

```
feld[2][3] = 10;
```

G.5 Eindimensionale Felder als Funktionsparameter

- ganze Felder können in C **nicht by-value** übergeben werden
- wird einer Funktion der Feldname als Parameter übergeben, kann sie in gleicher Weise wie der Aufrufer auf die Feldelemente zugreifen (und diese verändern!)
- bei der Deklaration des formalen Parameters wird die Feldgröße weggelassen
 - ▶ die Feldgröße ist automatisch durch den aktuellen Parameter gegeben
 - ▶ ggf. ist die Feldgröße über einen weiteren `int`-Parameter der Funktion explizit mitzuteilen
 - ▶ die Länge von Zeichenketten in `char`-Feldern kann normalerweise durch Suche nach dem `\0`-Zeichen bestimmt werden
- wird ein Feldparameter als `const` deklariert, können die Feldelemente innerhalb der Funktion nicht verändert werden (ANSI)

1 Beispiele

G.5 Eindimensionale Felder als Funktionsparameter

- Bestimmung der Länge einer Zeichenkette (*String*)

```
int strlen(const char string[])
{
    int i=0;
    while (string[i] != '\0') ++i;
    return(i);
}
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.9

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

G.6 Strukturen

1 Motivation

- Felder fassen Daten eines einheitlichen Typs zusammen
 - ungeeignet für gemeinsame Handhabung von Daten unterschiedlichen Typs
- Beispiel: Daten eines Studenten
 - Nachname `char nachname[25];`
 - Vorname `char vorname[25];`
 - Geburtsdatum `char gebdatum[10];`
 - Matrikelnummer `int matrnr;`
 - Übungsgruppennummer `short gruppe;`
 - Schein bestanden `char best;`
- Möglichkeiten der Repräsentation in einem Programm
 - einzelne Variablen → sehr umständlich, keine "Abstraktion"
 - Datenstrukturen

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.11

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

1 Beispiele (2)

G.5 Eindimensionale Felder als Funktionsparameter

- Konkateneriere Strings

```
void strcat(char to[], const char from[])
{
    int i=0, j=0;
    while (to[i] != '\0') i++;
    while ( (to[i++] = from[j++]) != '\0' )
        ;
}
```

- Funktionsaufruf mit Feld-Parametern

► als aktueller Parameter beim Funktionsaufruf wird einfach der Feldname angegeben

```
char s1[50] = "text1";
char s2[] = "text2";
strcat(s1, s2); /* → s1= "text1text2" */
strcat(s1, "text3"); /* → s1= "text1text2text3" */
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.10

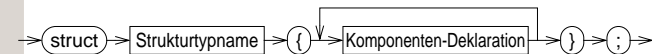
Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Deklaration eines Strukturtyps

G.6 Strukturen

- Durch eine Strukturtyp-Deklaration wird dem Compiler der Aufbau einer Datenstruktur unter einem Namen bekanntgemacht
 - deklariert einen neuen Datentyp (wie `int` oder `float`)

- Syntax



- Strukturtypname**

- beliebiger Bezeichner, kein Schlüsselwort
- kann in nachfolgenden Struktur-Definitionen verwendet werden

- Komponenten-Deklaration**

- entspricht normaler Variablen-Definition, aber keine Initialisierung!
- in verschiedenen Strukturen dürfen die gleichen Komponentennamen verwendet werden (eigener Namensraum pro Strukturtyp)

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.12

Reproduktion jeder Art oder Vervielfältigung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

2 Deklaration eines Strukturtyps (2)

G.6 Strukturen

■ Beispiele

```
struct student {
    char nachname[25];
    char vorname[25];
    char gebdatum[10];
    int matrnr;
    short gruppe;
    char best;
};
```

```
struct komplex {
    double re;
    double im;
};
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.13

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

4 Zugriff auf Strukturkomponenten

G.6 Strukturen

■ .-Operator

- $x.y$ \equiv Zugriff auf die Komponente y der Struktur x
- $x.y$ verhält sich wie eine normale Variable vom Typ der Strukturkomponenten y der Struktur x

■ Beispiele

```
struct komplex c1, c2, c3;
...
c3.re = c1.re + c2.re;
c3.im = c1.im + c2.im;

struct student stud1;
...
if (stud1.matrnr < 1500000) {
    stud1.best = 'y';
}
```

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

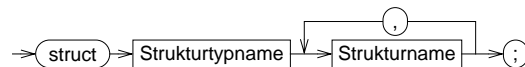
G.15

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

3 Definition einer Struktur

G.6 Strukturen

- Die Definition einer Struktur entspricht einer Variablen-Definition
 - ◆ Name der Struktur zusammen mit dem Datentyp bekanntmachen
 - ◆ Speicherplatz anlegen
- Eine Struktur ist eine Variable, die ihre Komponentenvariablen umfaßt
- Syntax



■ Beispiele

```
struct student stud1, stud2;
struct komplex c1, c2, c3;
```

- Strukturdeklaration und -definition können auch in einem Schritt vorgenommen werden

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.14

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

5 Initialisieren von Strukturen

G.6 Strukturen

- Strukturen können — wie Variablen und Felder — bei der Definition initialisiert werden
- Beispiele

```
struct student stud1 = {
    "Meier", "Hans", "24.01.1970", 1533180, 5, 'n'
};

struct komplex c1 = {1.2, 0.8}, c2 = {0.5, 0.33};
```

!!! Vorsicht

- bei Zugriffen auf eine Struktur werden die Komponenten durch die Komponentennamen identifiziert,
bei der Initialisierung jedoch nur durch die Position
 - ➔ potentielle Fehlerquelle bei Änderungen der Strukturtyp-Deklaration

C-Ing

Einführung in die Programmierung für Ingenieure — C
© Jürgen Kleinöder, Universität Erlangen-Nürnberg, IMMD IV, 1998

G-Felder.doc 1998-06-24 10.04

G.16

Reproduktion jeder Art oder Vervielfachung dieser Unterlagen, außer zu Lehrzwecken an der Universität Erlangen-Nürnberg, bedarf der Zustimmung des Autors.

6 Strukturen als Funktionsparameter G.6 Strukturen

- Strukturen können wie normale Variablen an Funktionen übergeben werden
 - ◆ Übergabesemantik: **call by value**
 - Funktion erhält eine Kopie der Struktur
 - auch wenn die Struktur ein Feld enthält, wird dieses komplett kopiert!
 - !!! Unterschied zur direkten Übergabe eines Feldes

- Strukturen können auch Ergebnis einer Funktion sein
 - Möglichkeit mehrere Werte im Rückgabeparameter zu transportieren

■ Beispiel

```
struct komplex komp_add(struct komplex x, struct komplex y) {
    struct komplex ergebnis;
    ergebnis.re = x.re + y.re;
    ergebnis.im = x.im + y.im;
    return(ergebnis);
}
```

8 Strukturen in Strukturen G.6 Strukturen

- Die Komponenten einer Struktur können wieder Strukturen sein

■ Beispiel

```
struct name {
    char nachname[25];
    char vorname[25];
};

struct student {
    struct name name;
    char gebdatum[10];
    int matrnr;
    short gruppe;
    char best;
};

struct prof {
    struct name pname;
    char gebdatum[10];
    int lehrstuhlnr;
};

struct student stud1;
strcpy(stud1.name.nachname, "Meier");
if (stud1.name.nachname[0] == 'M') {
    ...
}
```

7 Felder von Strukturen G.6 Strukturen

- Von Strukturen können — wie von normale Datentypen — Felder gebildet werden

■ Beispiel

```
struct student gruppe8[35];
int i;
for (i=0; i<35; i++) {
    printf("Nachname %d. Stud.: ", i);
    scanf("%s", gruppe8[i].nachname);
    ...
    gruppe8[i].gruppe = 8;

    if (gruppe8[i].matrnr < 1500000) {
        gruppe8[i].best = 'y';
    } else {
        gruppe8[i].best = 'n';
    }
}
```

9 Compilerabhängige Eigenschaften G.6 Strukturen

- Funktionsparameter
 - ◆ alte C-Compiler erlauben die Übergabe ganzer Strukturen als Funktionsparameter oder -rückgabewert nicht
 - dann nur Zeiger auf Strukturen möglich

■ Zuweisungen

- ◆ moderne C-Compiler erlauben die Zuweisung kompletter Strukturen

◆ Beispiel

```
struct komplex c1, c2;
c2 = c1;
```

- ◆ wenn der Compiler dies nicht erlaubt, komponentenweise zuweisen

```
c2.re = c1.re; c2.im = c1.im;
```

■ Namen von Strukturkomponenten

- ◆ alte C-Compiler erlauben nicht gleiche Komponentennamen in unterschiedlichen Strukturen