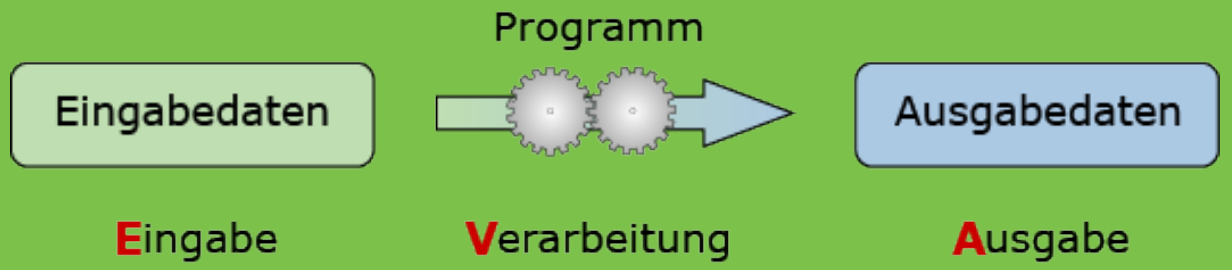
Hochschule Fulda
University of Applied Sciences

Programmierung 1 – Verbunddatentypen



Yvonne Jung

Wichtige Array-Operationen

Hochschule Fulda
University of Applied Sciences

- Wert an bestimmtem Index setzen oder holen
  - Wdh.: Felder sind Zusammenfassung von Elementen gleichen Typs
- Feld durchlaufen (d.h. traversieren)
  - …z.B. um ganzes Feld anzuzeigen oder um kleinstes Element darin zu finden
- Element in Feld suchen
  - Lineare Suche: Feld solange durchlaufen, bis Element gefunden wurde
    - Im *Worst Case* heißt das, bei Feldlänge *n*, alle *n* enthaltenen Elemente zu überprüfen
  - Binäre Suche: Deutlich schneller, aber setzt bereits sortiertes Feld voraus
- Feld sortieren
  - Auch hier verschiedene Algorithmen möglich, die sich je in Laufzeit oder Speicherbedarf unterscheiden (z.B. Selectionsort vs. Quicksort → *AlgoDat*)

Aufzählungsdatentyp

Hochschule Fulda
University of Applied Sciences

- Definition eines Aufzählungstyps mit Schlüsselwort “`enum`”
  - Bsp.:

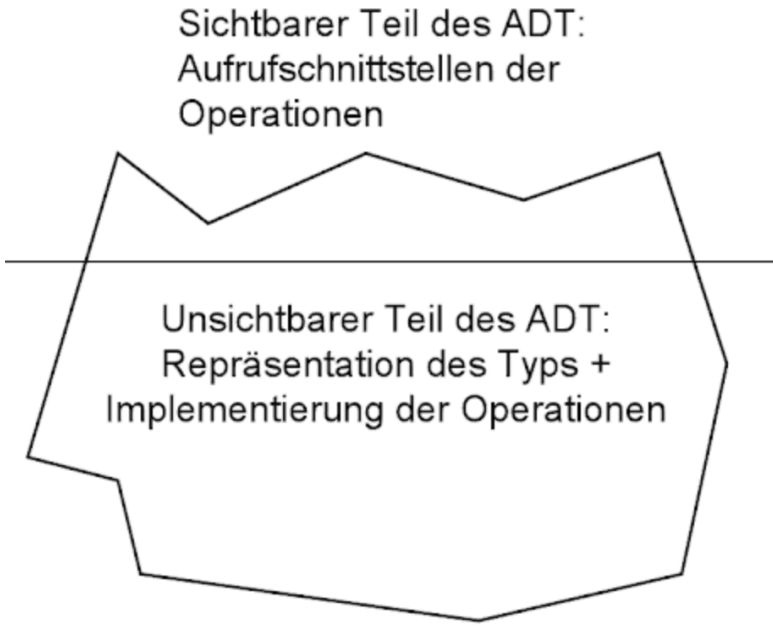
```
enum Primaerfarben
{
      ROT = 1, GRUEN = 2, BLAU = 4
};

enum Primaerfarben col = GRUEN;
```

- Elemente sind Optionen mit zugeordnetem Wert
  - Defaultmäßig hat erstes Element Wert 0 und nachfolgende Elemente haben je Wert des Vorgängers + 1
  - Einem Element kann aber anderer Integerwert zugewiesen werden (s.o.)
  - Werden typischerweise als Konstanten bzw. Flags verwendet

Hochschule Fulda
University of Applied Sciences

Abstrakte Datentypen

Datenstrukturen und Operationen



Abstrakter Datentyp (ADT)

Datenstruktur

Hochschule Fulda
University of Applied Sciences

- Bestimmte Art, Daten zu verwalten und miteinander zu verknüpfen
  - Ist mit bestimmten Operationen verbunden für Zugriff und Manipulation
- Ermöglicht, geeignet auf Daten zugreifen und sie ändern zu können
  - Gute Datenstrukturen unerlässlich für gute Algorithmen
  - Bei Entwurf konzeptionell erst mit abstraktem Datentyp arbeiten
- Zusammengesetzte Datentypen verbinden einfache Datentypen zu komplexeren Einheiten
  - Gruppieren Daten zu zusammengehöriger Gruppe (z.B. Adress-Datensätze)
  - Dazu werden u.a. mehrere, ggfs. verschiedenartige, Datentypen werden zu einem *Verbunddatentyp* (“`struct`” in C) zusammengefasst

Abstrakter Datentyp

Hochschule Fulda
University of Applied Sciences

- Beruht auf Angabe abstrakter Operationen auf abstrakt beschriebenen Daten
  - Beschreibung von Datenstrukturen mit Operationen unabhängig von Implementierung
    - → *Was* tut der ADT? **Nicht:** *Wie* tut er es?
  - Heißt abstrakt, weil keine konkrete Implementierung bzw. Programmiersprache angeben wird
    - → Abstrahiert also von Implementierung
- Operationen benötigen Signatur (→ Schnittstelle) und Semantik
  - ADT beschreibt nur, was Operationen tun (d.h. Semantik), aber nicht, wie sie es tun
  - Programmierer kennt nur Operationen zum Zugriff auf Daten, nicht deren interne Darstellung
    - Erst "Klassen" (C++, Java) erlauben, Daten u. Operationen zu einem Datentyp zusammenzufassen
- Erst bei Implementierung muss man entscheiden, wie Daten gespeichert werden
  - Datendarstellung getrennt von Beschreibungen der Operationen auf Daten
  - Art der internen Datenspeicherung hat Einfluss auf Laufzeit und Speicherverbrauch

Beispiel

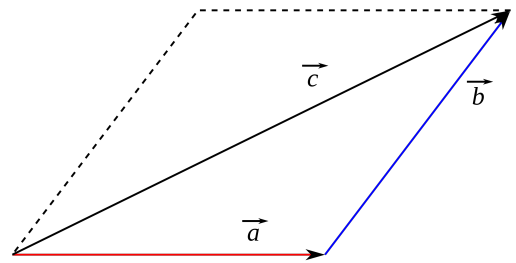
Hochschule Fulda
University of Applied Sciences

- ADT für Vektoren im $\mathbb{R}^n$ mit Operationen darauf
  - Zur Erinnerung: Vektoren sind *n*-dimensional (bei z.B. *n* = 3 also $\mathbb{R}^3$)
  - Alle Komponenten (z.B. $\vec{v} = (x \quad y \quad z)^T$) haben gleichen Wertebereich/Typ (z.B. $\mathbb{R}$)

```
#define DIM 3
typedef float Vec3[DIM];
```

```
void read_data(Vec3 v);
void show_data(Vec3 v);
void add(Vec3 res, Vec3 a, Vec3 b);
void sub(Vec3 res, Vec3 a, Vec3 b);
void mult(Vec3 res, Vec3 a, float s);
...
```

```
for (int i=0; i<DIM; i++)
    res[i] = a[i] + b[i];
```



Hochschule Fulda
University of Applied Sciences

Strukturen in C

Verbunddatentypen

Benutzername

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod

Kontakte

Foto

Bsp.: Profildaten in sozialem Netzwerk

Strukturen

Hochschule Fulda
University of Applied Sciences

- Zusammenfassung von Komponenten evtl. verschiedenen Typs zu einer Einheit
  - Durch Gruppierung von Variablen dient dies der Verwaltung/Organisation komplexer Daten
  - Z.B. Adress- oder Studierendendaten (Matrikelnummer, Name, Studiengang, Noten usw.)
- Werden vereinbart mit Schlüsselwort `struct`

```
struct [Bezeichner] {
    <Datentyp_a> komponente_1;
    ...
} [Strukturvariablen];
```

  - Strukturen werden meist *global* vor allen Funktionen definiert, sind damit überall bekannt
- Strukturkomponenten werden wie normale Variable vereinbart
  - Einzelne Komponenten werden nicht über Indizes, sondern über deren Namen angesprochen

Beispiele für Structs (1)

Hochschule Fulda
University of Applied Sciences

- Name eines Strukturtyps setzt sich zusammen aus *struct* + Bezeichner
- Mögliche Datenstruktur für Personen

```
struct Birthday {
    int day, month, year;
};
struct Person {
    char name[80];
    float height;
    struct Birthday birth;
};
```

- Strukturvariable anlegen

```
struct Person you;
```

- Zugriff auf Komponenten mit "."

```
you.height = 1.78f;
you.birth.year = 1992;
```

- Feld von Strukturtyp

```
struct Person group[100];
strcpy(group[0].name, "Fiona");
group[0].height = 1.67f;
group[0].birth.year = 2003;
```

Beispiele für Structs (2)

Hochschule Fulda
University of Applied Sciences

- Mögliche Datenstruktur für Punkt und Rechteck auf Bildschirm

```
struct Point {
    int x;
    int y;
};
struct Rectangle {
    struct Point p0;
    struct Point p1;
};
struct Point pnt = { 256, 256 };
struct Rectangle r = { {0, 0}, pnt };      // Initialisierung
```

Structs können ineinander verschachtelt sein, also direkt eingebettet in andere Struktur, und sogar anonym angegeben werden

- Rechteck `r` ausgeben

```
printf("((%d,%d)-(%d,%d)\n", r.p0.x, r.p0.y, r.p1.x, r.p1.y);
```

Zuweisung von Strukturen

Hochschule Fulda
University of Applied Sciences

- Strukturen können Strukturen gleichen Typs zugewiesen werden

```
struct Point p1, p2;
...
p1.x = 47; p1.y = 11;
p2 = p1;
```

- Dabei werden alle Komponenten kopiert
  - Nach Kopie haben beide Variablen gleichen Inhalt, aber verschiedene Stellen im Speicher
    - Strukturen kann man nicht direkt miteinander vergleichen, sondern nur komponentenweise
  - Strukturvariablen werden auch bei Parameterübergabe an Funktionen komplett kopiert
- Strukturen via *scanf()* befüllen

```
scanf("%d %d", &p1.x, &p1.y);
```

  - Im Fall von Zeichenketten auch bei Strukturelementen kein Adressoperator & davor

Alias auf C-Structs

Hochschule Fulda
University of Applied Sciences

- Wdh.: Mit `typedef` kann man nur neue Datentyp-Namen definieren
  - Aber keine neuen Datentypen
- Definition eines einfacheren Namens für einen Strukturtyp

```
typedef struct point Point;
```

  - Alternativ Definition schon direkt bei Struct-Angabe

```
typedef struct [point] {    //Bezeichner hier optional
    int x, y;
} Point;
```

  - Statt ***`struct`*** *`point`* kann man nun einfach `Point` schreiben
- Ist übersichtlicher (bessere Doku) und erfordert weniger Tipparbeit

Varianten

Hochschule Fulda
University of Applied Sciences

- Werden über Schlüsselwort “`union`” erzeugt
  - Verwendet, wenn Variable je nach Bedarf verschiedene Typen haben kann
  - Bsp.:

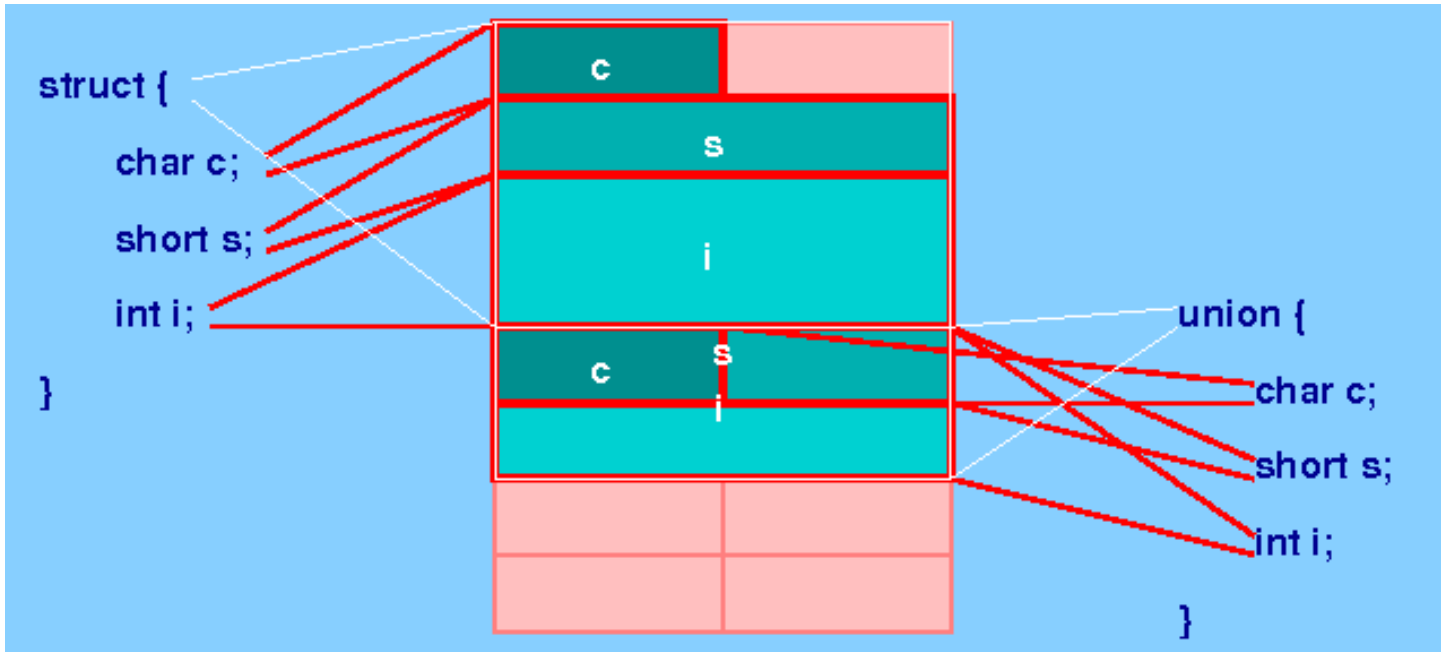
```
union Zahlcontainer
{
        char  charVal;
        int   intVal;
        float floatVal;
};
```

- Alle Komponenten teilen sich denselben Speicherbereich
  - Ändert man Komponente `intVal`, ändern sich auch `floatVal` und `charVal`
    - Beispiel-Union hat insges. nur 4 Byte (→ `sizeof(`*`union Zahlcontainer`*`)`)
  - Oft in Kombination mit *`enum`*, das besagt, welche Komponente gerade gültig ist

Speicherung von Unions

Hochschule Fulda
University of Applied Sciences

- Elemente in `union` teilen sich Speicherplatz
  - Speichern der Komponenten in `union` nicht nacheinander wie in `struct`, sondern „übereinander“, an gleichem Speicherplatz, daher nur je eine gültig



Vielen Dank!

Noch Fragen?