13. Datenstrukturen

Hochschule RheinMain

http://www.denkschatz.de/zitate/Willy-Brandt/Jetzt-wachst-zusammen-was-zusammengehort

Notizen

Datenbasis für die Beispiele

Hochschule RheinMain

Auf den Internetseiten des Deutschen Fussball-Bundes findet man eine Tabelle mit der Bilanz aller Fußballspiele der deutschen Nationalmannschaft, die wir zur weiteren Verarbeitung in eine Textdatei (`Laenderspiele.txt`) geschrieben haben:

Bilanz

Land	Spiele	gew.	unent.	verl.	Tore	Erstes Spiel	Letztes Spiel
Ägypten	1	0	0	1	1:2	28.12.1958	28.12.1958
Albanien	14	13	1	0	38:10	08.04.1967	06.06.2001
Algerien	2	0	0	2			
Argentinien	20	6	5	9			
Armenien	2	2	0	0			
Aserbaidschan	4	4	0	0			
Australien	4	3	0	1			
Belgien	25	20	1	4			
Böhmen-Mähren	1	0	1	0			
Bolivien	1	1	0	0			
Bosnien-Herzegowina	2	1	1	0			

```
Aegypten 1 0 0 1 1:2 28.12.1958 28.12.1958
Albanien 14 13 1 0 38:10 08.04.1967 06.06.2001
Algerien 2 0 0 2 1:4 01.01.1964 16.06.1982
Argentinien 20 6 5 9 28:28 08.06.1958 15.08.2012
Armenien 2 2 0 0 9:1 09.10.1996 10.09.1997
Aserbaidschan 4 4 0 0 15:2 12.08.2009 07.06.2011
Australien 4 3 0 1 12:5 18.06.1974 29.03.2011
Belgien 25 20 1 4 58:26 16.05.1910 11.10.2011
Boehmen-Maehren 1 0 1 0 4:4 12.11.1939 12.11.1939
Bolivien 1 1 0 0 1:0 17.06.1994 17.06.1994
Bosnien-Herzegowina 2 1 1 0 4:2 11.10.2002 03.06.2010
Brasilien 21 4 5 12 24:39 05.05.1963 10.08.2011
Bulgarien 21 16 2 3 56:24 20.10.1935 21.08.2002
Chile 6 4 0 2 11:7 23.03.1960 20.06.1982
China 2 1 1 0 2:1 12.10.2005 29.05.2009
Costa-Rica 1 1 0 0 4:2 09.06.2006 09.06.2006
Daenemark 26 15 3 8 53:36 06.10.1912 17.06.2012
DDR 1 0 0 1 0:1 22.06.1974 22.06.1974
Ecuador 2 2 0 0 7:2 20.06.2006 29.05.2013
Elfenbeinkueste 1 0 1 0 2:2 18.11.2009 18.11.2009
England 32 11 6 15 41:67 20.04.1908 27.06.2010
Estland 3 2 0 1 11:3 08.1935 29.  1939
```

Laenderspiele.txt

Eine Zeile in dieser Datei bildet einen zusammengehörigen Datensatz, den man als Ganzes verarbeiten (zum Beispiel einlesen, ausgeben, ändern) will. Mit unseren bisherigen Mitteln ist das nicht möglich.

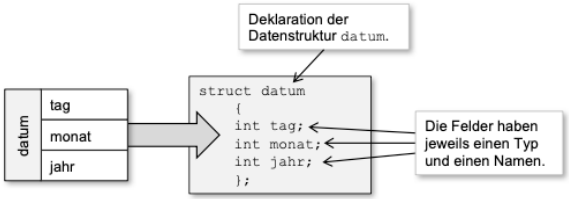
Notizen

Deklaration von Datenstrukturen

Hochschule RheinMain

In den beiden letzten Spalten der Länderspieltabelle stehen die Kalenderdaten für das erste und das letzte Spiel gegen die jeweils andere Nation. Wir wollen Tag, Monat und Jahr eines Datums so zusammenfassen, dass man ein Datum als Ganzes behandeln, aber auch auf gezielt auf Tag, Monat und Jahr zugreifen kann.

Dazu deklarieren wir eine Datenstruktur:



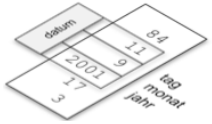
Erstes Spiel	Letztes Spiel
28.12.1958	28.12.1958
08.04.1967	06.06.2001
01.01.1964	16.06.1982
08.06.1958	15.08.2012
09.10.1996	10.09.1997
12.08.2009	07.06.2011
18.06.1974	29.03.2011
16.05.1910	11.10.2011
12.11.1939	12.11.1939
17.06.1994	17.06.1994
[illegible]	03.06.2010

Notizen

Datenstrukturen sind keine Daten

Hochschule RheinMain

Mit der Deklaration einer Datenstruktur entstehen keine Daten und kein Code. Eine Datenstruktur ist nur Schablone, durch die wir auf unsere Daten blicken wollen. Die Schablone strukturiert die Daten.

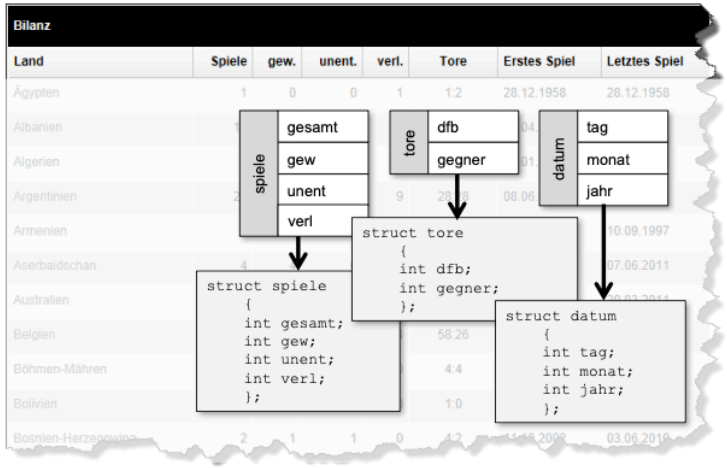


Die elementaren Datentypen (`char`, `int`, `float`, `double`...) sind der Rohstoff, aus dem Datenstrukturen zusammengesetzt werden können.

Notizen

Weitere Datenstrukturen für die Spielbilanz

Hochschule RheinMain



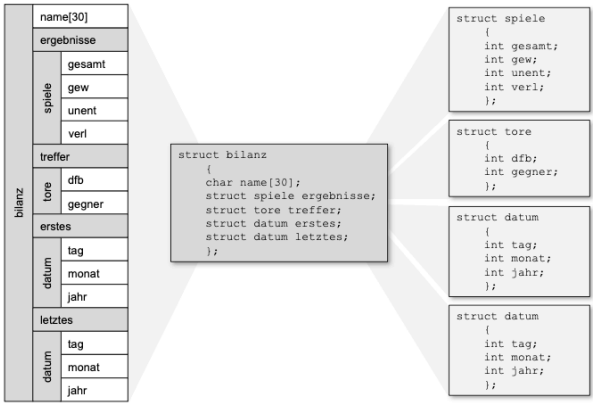
Nach Bedarf können aus allen Grunddatentypen Datenstrukturen zusammengestellt werden, auch wenn wir es in unserem Beispiel nur mit ganzen Zahlen zu tun haben.

Notizen

Komplexere Strukturen

Hochschule RheinMain

Datenstrukturen können Strukturen und Arrays enthalten. Zur Modellierung einer Zeile der Länderspieltabelle greifen wir auf die bereits deklarierten Teilstrukturen (`spiele, tore, datum`) zurück und fügen noch einen Array von 30 Zeichen für den Namen des Landes hinzu:



Notizen

Bezeichner in Datenstrukturen

Hochschule RheinMain

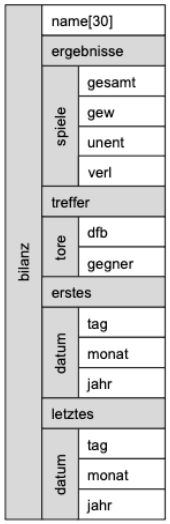
Alles in einer Datenstruktur hat einen Namen. Grundsätzlich gibt es zwei verschiedene Arten von Namen:

Dazu deklarieren wir eine Datenstruktur:

- Struktur-Namen (in der Grafik senkrecht geschrieben), wie `bilanz` oder `datum`. Mit diesen Namen werden neue Strukturen eindeutig benannt.
- Feld-Namen (in der Grafik waagerecht geschrieben), wie `monat` oder `treffer`. Diese Namen dienen zum Zugriff auf die Felder einer Datenstruktur

Die Struktur `bilanz` enthält zum Beispiel unter dem Namen `ergebnisse` eine Struktur `spiele`.

Die Struktur `datum` ist zweimal in der Struktur `bilanz` vorhanden. Auf das eine Datum kann unter dem Namen `erstes`, auf das zweite unter dem Namen `letztes` zugegriffen werden.

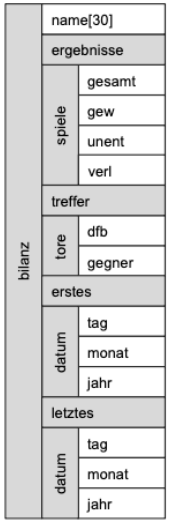


Notizen

Bezeichner in Datenstrukturen

Hochschule RheinMain

Wie ein Zugriff auf die Daten konkret aussieht, werden wir später sehen. Noch gibt es ja gar keine Daten sondern nur Schablonen mit Struktur- und Zugriffsinformationen.

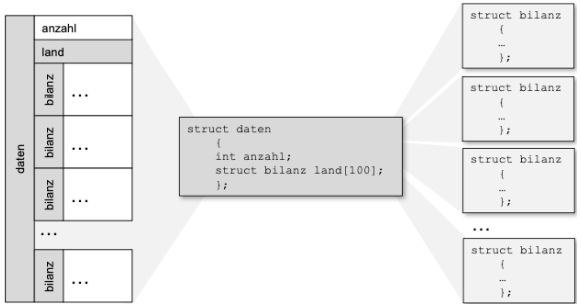


Notizen

Gesamtmodell

Hochschule RheinMain

Um die Tabelle mit den Länderspielbilanzen als Ganzes zu modellieren, werden wir jetzt noch einen Array von ausreichend vielen (100) Bilanzen erstellen und zusätzlich speichern, wie viele Einträge in diesem Array gültig sind.



Beachten Sie, dass die Arrays in diesem Beispiel auf die zu erwartende Maximallast (maximal 29 Buchstaben im Ländernamen, maximal 100 verschiedene Länder) ausgelegt sind. Das ist eine Beschränkung, von der wir uns später befreien werden.

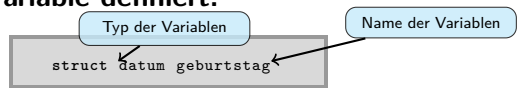
Notizen

Variablendefinition

Hochschule RheinMain

Durch die Deklaration einer Datenstruktur wird nur ein neuer Datentyp eingeführt. Üblicherweise findet man Datenstruktur-Deklarationen in Headerdateien, die dann von allen Quelldateien, die diese Datenstrukturen verwenden wollen, inkludiert werden. Werden Datenstruktur-Deklarationen nur in einer einzigen Quelldatei benötigt, können sie auch dort, typischerweise am Anfang der Datei, stehen.

Konkrete Daten einer bestimmten Struktur erhält man erst, wenn man eine Variable definiert:

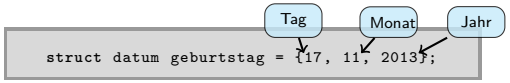


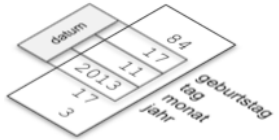
Notizen

Variablendefinition

Hochschule RheinMain

Jetzt ist ein konkretes Datum (`geburtstag`) entstanden, das auch schon bei der Definition mit Werten gefüllt werden kann:



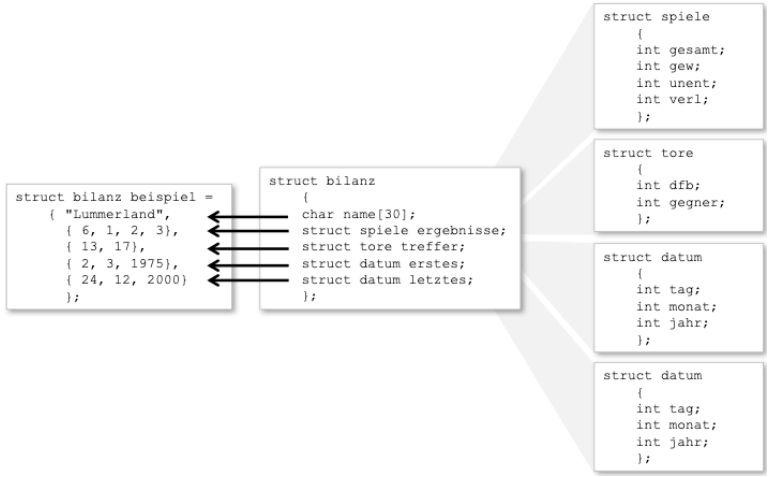


Notizen

Def. und Init. komplexer Strukturvariablen

Hochschule RheinMain

Auch komplexe, verschachtelte Strukturen können angelegt und initialisiert werden. Man folgt einfach der durch die Schablone vorgegebenen Struktur.

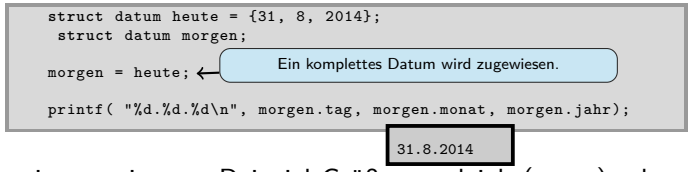


Notizen

Zuweisung von Datenstrukturen

Hochschule RheinMain

Die Werte einer Variablen können einer anderen Variablen zugewiesen werden, egal, ob die Variablen nur auf einem einfachen Datentyp oder einer komplexen Struktur basieren. Wichtig ist, dass bei einer Zuweisung auf der linken und rechten Seite des Gleichheitszeichens der gleiche Datentyp steht.



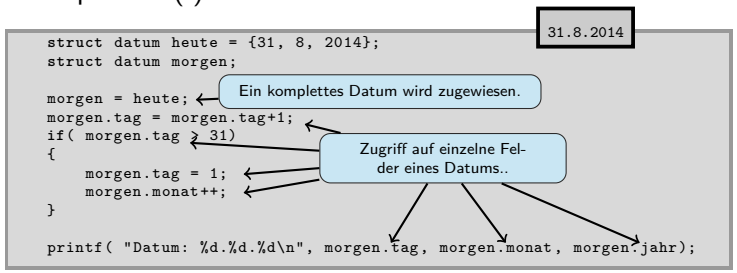
Operationen wie zum Beispiel Größenvergleich ($<$, $>$) oder arithmetische Operationen kann man auf Datenstrukturen nicht ausführen (da müssen wir uns noch bis zur objektorientierten Programmierung gedulden). Wie sollte der Compiler auch wissen, wie etwa der Vergleich zweier Kalenderdaten, im Sinne eines Vorher-Nachher-Vergleichs, durchgeführt werden sollte.

Notizen

Direktzugriff auf die Felder einer Datenstruktur

Hochschule RheinMain

Zum direkten Zugriff auf die Felder einer Datenstruktur dient der Punkt-Operator (.):

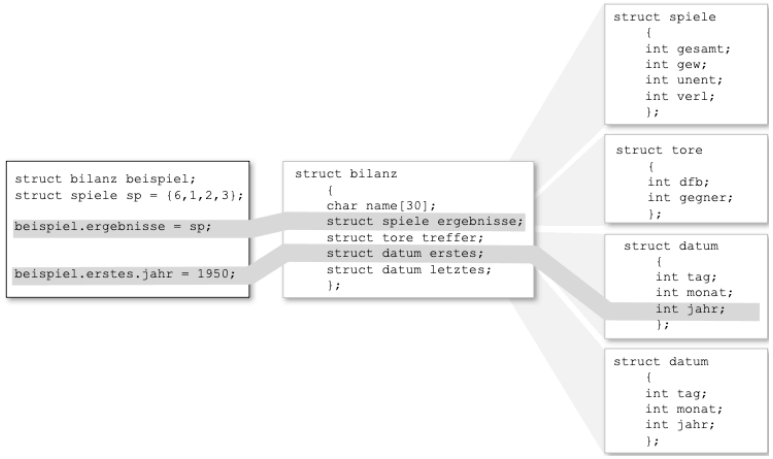


Notizen

Zugriff in verschachtelte Datenstrukturen

Hochschule RheinMain

Schritt für Schritt kann man mit dem Punkt-Operator in eine Datenstruktur hineinzoomen, bis man auf dem Level angekommen ist, auf dem man arbeiten möchte, egal, wie tief die Strukturen verschachtelt sind

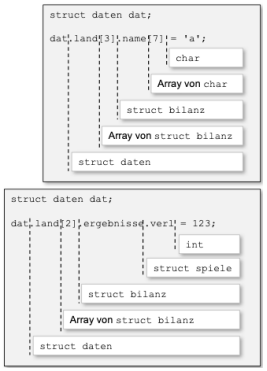


Notizen

Datentypen beim Zugriff

Hochschule RheinMain

Wichtig ist, immer im Blick zu behalten, welchen Datentyp man auf welcher Zugriffsstufe jeweils erhält, damit man weiß, welche Operationen man auf dem jeweiligen Level ausführen kann.




```
struct datum
{
    int tag;
    int monat;
    int jahr;
};
struct spiele
{
    int gesamt;
    int gew;
    int unent;
    int verl;
};
struct tore
{
    int dfb;
    int gegner;
};
struct bilanz
{
    char name[30];
    struct spiele ergebnisse;
    struct tore treffer;
    struct datum erstes;
    struct datum letztes;
};
struct daten
{
    int anzahl;
    struct bilanz land[100];
};
```


Notizen

Indirektzugriff auf Datenstrukturen

Hochschule RheinMain

Wir können auch Zeiger auf Datenstrukturen anlegen, so wie wir bereits Zeiger auf die Grunddatentypen angelegt hatten:

```
struct datum *pointer;
```
← Ein Zeiger auf ein Datum

Bei `pointer` handelt es sich nicht um eine Datenstruktur mit Feldern `tag`, `monat` und `jahr`, sondern `pointer` ist ein Zeiger, der die Adresse einer solchen Datenstruktur enthält.

Der Zeiger ist unbrauchbar, solange ihm nicht die Adresse einer konkreten Datenstruktur zugewiesen wird:

```
struct datum geburtsdatum;
struct datum *pointer;

pointer = &geburtsdatum;

(*pointer).tag = 17;
(*pointer).monat = 11;
(*pointer).jahr = 2013;
```
← Noch haben geburtsdatum und pointer nichts miteinander zu tun.

← Der Zeiger erhält einen Adresswert.

← Über den Zeiger werden Werte in die referenzierte Datenstruktur eingetragen

Mit dem Adressoperator (`&`) kann man die Adresse eine Variablen ermitteln und mit dem Dereferenzierungsoperator (`*`) kann man über eine Adresse auf eine Variable zugreifen.

Notizen

Der Points-Operator

Hochschule RheinMain

Ist `p` ein Zeiger auf eine Datenstruktur und `x` ein Feld dieser Datenstruktur, so lässt sich auf das Feld mit den beiden geichwertigen Ausdrücken `(*p).x` bzw. `p->x` zugreifen.

Beide Ausdrücke sind dabei als R-Value und L-Value – also sowohl auf der rechten als auch auf der linken Seite einer Zuweisung – geeignet.
Den Ausdruck `p->x` lesen wir als ≫p points x≪

Damit kann man den Strukturzugriff eleganter formulieren:

Zugriff mit *-Operator

```
struct datum geburtsdatum;
struct datum *pointer;

pointer = &geburtsdatum;

(*pointer).tag = 17;
(*pointer).monat = 11;
(*pointer).jahr = 2013;
```

Zugriff mit Points-Operator

```
struct datum geburtsdatum;
struct datum *pointer;

pointer = &geburtsdatum;

pointer->tag = 17;
pointer->monat = 11;
pointer->jahr = 2013;
```

Den wahren Wert von Zeigern erkennt man aber erst im Zusammenhang mit Funktionen und dynamischen Datenstrukturen.

Notizen

Memory-mapped I/O über Zeiger

Hochschule RheinMain

Register in Strukturen zusammenfassen

```
typdef struct {
   volatile uint8_t pin; /* Pegelregister       */
   volatile uint8_t ddr; /* Data direction reg. */
   volatile uint8_t port;/* Ausgaberegister     */
} atmega_port;

#define AVR_PORTA ((atmega_port*)0x39)
#define AVR_PORTB ((atmega_port*)0x36)
#define AVR_PORTC ((atmega_port*)0x33)
#define AVR_PORTD ((atmega_port*)0x30)
```

Abbilden einer (logisch zusammengehörigen) Gruppe von Registern durch eine Datenstruktur

Gleichartige E/A-Geräte (hier: ATMEGA Ports A ...D) werden durch die gleiche Datenstruktur mit jeweils anderer Adresse beschrieben

Notizen

Memory-mapped I/O über Zeiger

Hochschule RheinMain

Damit: Blink-Routine (vgl. 10.8):

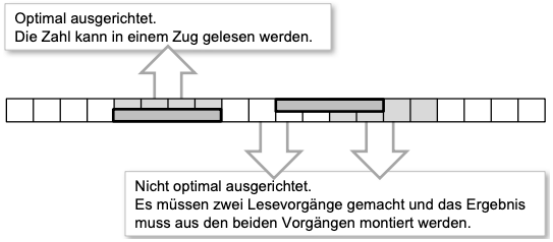
```
void blink(int led, int times)
{
  AVR_PORTC->ddr  = (1<<6)|(1<<5);
  while(times--) {
    AVR_PORTC->port = (led & 3) << 5;
    _delay_ms(250);
    AVR_PORTC->port = 0;
    _delay_ms(250);
  AVR_PORTC->ddr = ( 1 << 7 );
  while( wie_oft-- )
  {
    AVR_PORTC->port |= ( 1 << 7);   // LED an
    _delay_ms( 250 );
    AVR_PORTC->port &= ~( 1 << 7 ); // LED aus
    _delay_ms ( 250 );
  }
  }
}
```

Notizen

Alignment

Hochschule RheinMain

Intern werden Datenstrukturen so abgelegt, dass der Rechner optimal zugreifen kann. Wenn ein Rechner etwa eine 4-Byte Integer-Darstellung hat, greift er den Speicher in 4-Byte Blöcken ab und kann auf eine 4-Byte Zahl besonders effizient zugreifen, wenn sie auf einer durch 4 teilbaren Adresse beginnt.

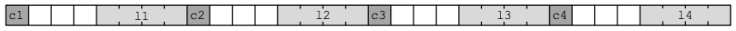


Notizen

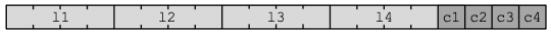
Alignment

Hochschule RheinMain

Diese als **Alignment** bezeichnete Strategie zur Ausrichtung von Daten im Speicher führt dazu, dass im ersten Fall



sehr viel mehr Speicher benötigt wird als im zweiten.




```
struct test1
    {
    char c1;
    long l1;
    char c2;
    long l2;
    char c3;
    long l3;
    char c4;
    long l4;
    };
```

```
struct test2
    {
    long l1;
    long l2;
    long l3;
    long l4;
    char c1;
    char c2;
    char c3;
    char c4;
    };
```

Sie können ihre Datenstrukturen sehr einfach optimieren, indem Sie die Datenfelder absteigend nach Größe anordnen.

In der Programmlogik besteht kein Unterschied zwischen den beiden Varianten der Datenstruktur.

Notizen