Programmierung 1

Primitive Datentypen

Monika Schak

Woche 4

11. November 2024

Hochschule Fulda
University of Applied Sciences

Wiederholung

- Drei Arten von Schleifen:
  - Zählschleife: `for` mit Initialisierung, Abbruchbedingung und Schritt im Schleifenkopf
  - Kopfgesteuerte Schleifen: `while`
  - Fußgesteuerte Schleifen: `do-while`
- Zwei zusätzliche Kontrollmöglichkeiten:
  - Abbruch von Schleifen mit `break`
  - Abbruch von Iterationen mit `continue`

Hochschule Fulda
University of Applied Sciences

Primitive Datentypen in C

	Verwendung	OS 16Bit	OS 32Bit	OS 64Bit
`char`[1]	ASCII-Zeichen	1 Byte	1 Byte	1 Byte
`short`[1]	Ganzzahl	2 Byte	2 Byte	2 Byte
`int`[1]	Ganzzahl	2 Byte	4 Byte	4 Byte
`long`[1]	Ganzzahl	4 Byte	4 Byte	4/8 Byte
`float`	Gleitkommazahl	4 Byte	4 Byte	4 Byte
`double`	Gleitkommazahl	8 Byte	8 Byte	8 Byte
`bool`[2]	Wahrheitswert	$\geq$ 1 Byte	$\geq$ 1 Byte	$\geq$ 1 Byte

1 == sizeof(`char`) <= sizeof(`short`) <= sizeof(`int`) <= sizeof(`long`)

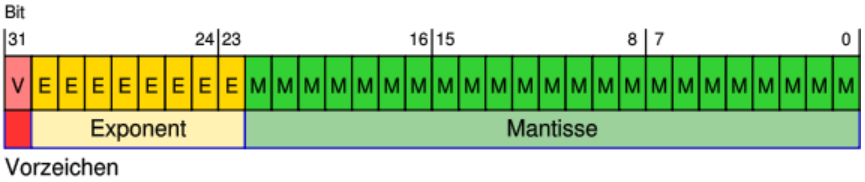
[1] Datentyp kann mit vorangestelltem `unsigned` vorzeichenlos werden.

[2] Nur verfügbar, wenn man `stdbool.h` einbindet.

Hochschule Fulda
University of Applied Sciences

Datentyp float

- Für Gleitpunktzahlen gibt es den Datentyp `float`
- Die Größe eines `float`s beträgt 4 Byte: Der Wertebereich geht daher grob von $\pm 1.2 * 10^{-38}$ bis $\pm 3.4 * 10^{+38}$
- Interne Darstellung basiert auf Zerteilung in Vorzeichen, Mantisse und Exponent
- Für Ein- und Ausgabe von Floating-Point-Zahlen gibt es den Platzhalter `%f`




```
float pi = 3.1415926f, eps = 1E-6f;
printf("Pi = %f, Epsilon = %f\n");
```

Hochschule Fulda
University of Applied Sciences

Typumwandlungen

- Nötig, wenn zwei Operanden in einem Ausdruck verschiedene Typen haben
- Typumwandlungen werden wenn nötig und möglich implizit durchgeführt
  - `int a = 12, b = 18;`
    `float f = b / a;`
    `printf("%f\n", f);`
  - Problem: `f` hat den Wert 1.0, weil die Typumwandlung erst nach der Division erfolgt
- Explizite Typumwandlung (**Type Cast**) mit Cast-Operator: in Klammern gesetzte Typangabe vor Ausdruck
  - Lösung: `b` wird explizit auf `float` gecastet und damit implizit auch `a`, also ist `f` nun 1.5
  - `int a = 12, b = 18;`
    `float f = (float) b / a;`
    `printf("%f\n", f);`

Hochschule Fulda
University of Applied Sciences

Konstanten

- In C ursprünglich mit Hilfe von Präprozessordirektiven: reine Ersetzung von Zeichensetzung vor der Kompilierung
  - Werden meist direkt nach den Include-Direktiven angegeben
  - Präprozessor geht Code vor eigentlicher Übersetzung durch und reagiert auf `#`
  - Bindet Include-Dateien in Code ein und fügt Definitionen durch Textersatz ein
  - Beispiel: `#define PI 3.14159265358979328462643383279502888`
  - Achtung: Kein Semikolon am Ende, da bei der Benutzung reiner Textersatz stattfindet
  - Konstanten werden typischerweise in Großbuchstaben geschrieben
- Typischer definierbar über Typ-Qualifizierer `const`
  - Durch zusätzliches Schlüsselwort `const` kann eine Variable nur gelesen werden
  - Beispiel: `const double Pi = 3.14159265358979328462643383279502888;`

Hochschule Fulda
University of Applied Sciences

Defintion von Typnamen

- Schlüsselwort `typedef`
- Es wird kein neuer Datentyp geschaffen, sondern neuer Name festgelegt
- Typdefinitionen stehen am Anfang des Codes außerhalb von Funktionen (nach Präprozessor-Anweisungen)
- Beispiel: `typedef double Real;` → Typ `double` heißt dann `Real`
- Vorteil:
  - Einfaches, durchgängiges Ändern des Typs von Variablen
  - ggf. besser lesbare Typnamen verwenden, die an die Aufgabenstellung angepasst sind
  - Verwendung zusammengesetzter Datentypen → später!

Hochschule Fulda
University of Applied Sciences

Formatzeichen

`%c`	Einzelzeichen, Typ `char`
`%s`	Zeichenkette, d.h. `char[]`
`%d`	Ganzzahl als Dezimalzahl, Typ `int`
`%x`	Ganzzahl als vorzeichenlose Hexadeizmalzahl
`%p`	Speicheradresse (Zeigerwert)
`%hd`	Ganzzahltyp `short int`
`%ld`	Ganzzahltyp `long int`
`%lld`	Ganzzahltyp `long long`
`%u`	vorzeichenlose Ganzzahl: `unsigned int`
`%llu`	vorzeichenlos Ganzzahl: `unsigned long long`
`%f`	Gleitkommazahl als `float`
`%lf`	Gleitkommazahl als `double`
`%e`	Gleitkommazahl in Exponentendarstellung

Hochschule Fulda
University of Applied Sciences

Wichtige Steuerzeichen

`\n`	Newline (Zeilenumbruch)
`\r`	Carriage Return (Wagenrücklauf)
`\t`	Tabulator
`\b`	Backspace
`\0`	Endezeichen in Strings
`\'`	Einfaches Anführungszeichen ’
`\"`	Doppeltes Anführungszeichen “
`%%`	Prozentzeichen %
`\\`	Escapezeichen \

Hinweis: Textdateien unter Unix, Linux und MacOS verwenden für den Zeilenumbruch die Escape-Sequenz `\n`, während unter Windows `\r\n` dafür verwendet wird.

Hochschule Fulda
University of Applied Sciences

Zusammengesetzte Operatoren

- C erlaubt eine abgekürzte Schreibweise für bestimmte Zuweisungen

+=	`x += <Ausdruck>`	←	`x = x + <Ausdruck>`
-=	`x -= <Ausdruck>`	←	`x = x - <Ausdruck>`
*=	`x *= <Ausdruck>`	←	`x = x * <Ausdruck>`
/=	`x /= <Ausdruck>`	←	`x = x / <Ausdruck>`
%=	`x %= <Ausdruck>`	←	`x = x % <Ausdruck>`

- Beispiel: `count += 2` ist gleichwertig mit `count = count + 2`
- Achtung: Funktioniert in C nicht für boolesche Ausdrücke! Geht aber für korrespondierende Bitoperationen ($\rightarrow$ später!)

Hochschule Fulda
University of Applied Sciences

Auswertungsreihenfolge

- Assoziativität: Besagt, in welcher Reihenfolge pro Zeile ausgewertet wird, d.h. von links nach rechts oder von rechts nach links
- Präzedenz: Beschreibt, welcher Operator zuerst ausgewertet wird, d.h. Operatoren mit kleinster Präzedenz werden zuerst ausgewertet

Hochschule Fulda
University of Applied Sciences

Auswertungsreihenfolge

Präzedenz	Operator	Beschreibung	Assoziativität
1	(Postfix)++	Postfix-Inkrement	$\rightarrow$
	(Postfix)$--$	Postfix-Dekrement	
	()	Funktionsaufruf	
	[]	Indizierung (Arrays)	
	->	Elementzugriff (mit Pointer)	
	.	Elementzugriff	
2	++(Präfix)	Präfix-Inkrement	$\leftarrow$
	$--$(Präfix)	Präfix-Dekrement	
	+, $-$	Vorzeichen	
	!	logisches NICHT	
	$\sim$	bitweises NICHT	
	(type)	Type-Cast	
	*	Dereferenzierung (Pointer)	
	&	Adresse	
	sizeof	Speichergröße	

Auswertungsreihenfolge

Präzedenz	Operator	Beschreibung	Assoziativität
3	* / %	Multiplikation Division Rest (Modulo)	$\rightarrow$
4	+ $-$	Addition Subtraktion	$\rightarrow$
5	$<<$ $>>$	Links-Shift Rechts-Shift	$\rightarrow$
6	<, <= >, >=	Kleiner (gleich) Größer (gleich)	$\rightarrow$
7	== !=	Gleich Ungleich	$\rightarrow$
8	&	Bitweises UND	$\rightarrow$

Auswertungsreihenfolge

Präzedenz	Operator	Beschreibung	Assoziativität
9	^	Bitweises XOR	$\rightarrow$
10	|	Bitweises ODER	$\rightarrow$
11	&&	Logisches UND	$\rightarrow$
12	||	Logisches ODER	$\rightarrow$
13	? :	Ternärer Operator	$\leftarrow$
14	= +=, -=, *= /=, %=, «=, »= &=, ^=, |=	Zuweisung Zusammengesetzte Zuweisung	$\leftarrow$
15	,	Komma-Operator	$\rightarrow$

Übung

Gib für jede Zeile die richtige Auswertungsreihenfolge sowie die neuen Werte der Variablen an:

```
int a, b = 5, c, d = 20;
a = b / 2;
c = b % 2;
b = 1 - --b;
b *= -3;
d %= 3;
c += b * d + 4;
a = --b + d++;
a = 0; b = 2; c = 3; d = 4;
a = (b + 2) * 2 * c + 1;
a = ++b * d++ * ++c * (-1);
```

Hochschule Fulda
University of Applied Sciences

Problematik Inkrement/Dekrement

- Vorsicht bei mehreren Inkrementen/Dekrementen in einer Zeile, bzw. bei Verwendung von Inkrement/Dekrement in Verbindung mit arithmetischen Ausdrücken!

```
int x = 4;
printf("%d", x + ++x);
```

Hochschule Fulda
University of Applied Sciences

Problematik Inkrement/Dekrement

- Vorsicht bei mehreren Inkrementen/Dekrementen in einer Zeile, bzw. bei Verwendung von Inkrement/Dekrement in Verbindung mit arithmetischen Ausdrücken!

```
int x = 4;
printf("%d", x + ++x);
```

  - Compilerabhängig unterschiedliche Behandlung (C17)
  - Sequenz der Berechnung im Standard nicht komplett festgelegt
  - GCC $\rightarrow$ Ergebnis: 10
  - CLANG $\rightarrow$ Ergebnis: 9

Hochschule Fulda
University of Applied Sciences