Einführung in die Programmierung

Grundbegriffe

Prof. Dr. Peter Jüttner

Inhalt

- **Grundbegriffe**
  - Grundstruktur eines C Programms
  - Variable
  - Konstante
  - Typen
  - Ein-/Ausgabe
  - Operatoren
  - Kontrollstrukturen
- **Komplexe Elemente**
  - Weitere Typen
  - Pointer
  - Funktionen

Ein-/Ausgabe

Hochschule Deggendorf

Grundsätzliches

- mittels vordefinierter Bibliotheksfunktionen (Unterprogramme)
- nicht im eigentlichen C-Sprachumfang
- standardisiert, d.h. in allen C-Compilern für PC-Anwendungen verfügbar
- (eher) nicht für embedded Compiler

Ein-/Ausgabe

Hochschule Deggendorf

Ausgabe auf Bildschirm mittels printf

```
#include <stdio.h>

int main(void)
{
      #define PI 3.14

      float radius = 5;
      float kreisumfang = 2*radius*PI;
      float kreisflaeche = radius*radius*PI;

      printf("Kreisumfang bei Radius 5  :   %.2f\n", kreisumfang);
      printf("Kreisflaeche bei Radius 5   :   %.2f\n", kreisflaeche);


};
```

Ein-/Ausgabe

Hochschule Deggendorf

Ausgabe auf Bildschirm mittels printf

- grundsätzliche Struktur

  printf("<Format>", <auszugebende Daten>)

  wobei <Format> in der Form

  %[Flag][Breite][.Präzision][Präfix]Typ

  angegeben wird.

  Die in [...] angegebenen Formatteile können weggelassen werden.

Ein-/Ausgabe

Hochschule Deggendorf

Ausgabe auf Bildschirm mittels printf

%[Flag][Breite][.Präzision][Präfix]**Typ**

Typ-angabe legt Ausgabetyp (d.h. Ausgabeformat) fest, u.a.:

%d %i Dezimalzahl (mit Vorzeichen)

%o Oktalzahl

%x %X Hexadezimalzahl (klein / gross)

%u Dezimalzahl (ohne Vorzeichen)

%c Buchstabe (Character)

%s Zeichenkette (String)

Ein-/Ausgabe

Hochschule Deggendorf

Ausgabe auf Bildschirm mittels printf

%[Flag][Breite][.Präzision][Präfix]**Typ**

Typ-angabe legt Ausgabetyp (d.h. Ausgabeformat) fest, u.a.:

%f Gleitkommazahl (Kommadarstellung)

%e %E Gleitkommazahl (Exponentialdarstellung)

%g %G Double (kürzere Form von %e oder %f, bzw. %E oder %f)

%p Zeiger (Pointer)

%n Anzahl auszugebender Zeichen

Ein-/Ausgabe

Hochschule Deggendorf

Ausgabe auf Bildschirm mittels printf

%[**Flag**][Breite][.Präzision][Präfix]Typ

Flag-angabe, optionales – Zeichen legt linksbündige Ausgabe fest

%[Flag][**Breite**][.Präzision][Präfix]Typ

die Zahl Breite legt die minimale Breite des Ausgebefeldes fest

%[Flag][Breite][**.Präzision**][Präfix]Typ

die Zahl Präzision legt die Anzahl der Nachkommastellen fest

Ein-/Ausgabe

Hochschule Deggendorf

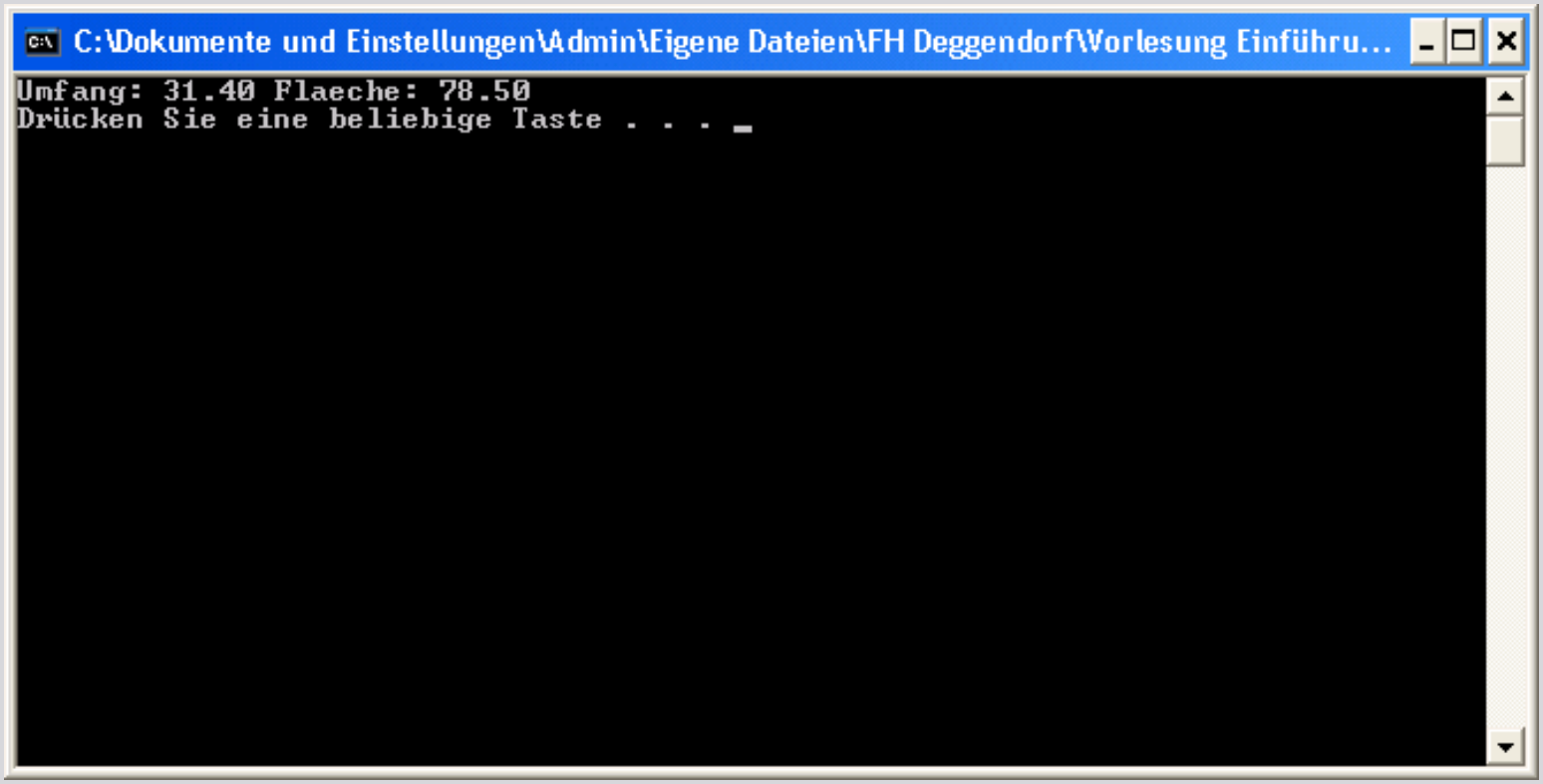
Ausgabe mehrer Werte auf Bildschirm mittels printf

```
#include <stdio.h>

int main(void)
{
	#define PI 3.14

	float radius = 5;
	float ku = 2*radius*PI;
	float kf = radius*radius*PI;

	printf("Umfang: %.2f Flaeche: %.2f\n", ku, kf);

};
```

Ein-/Ausgabe

Hochschule Deggendorf

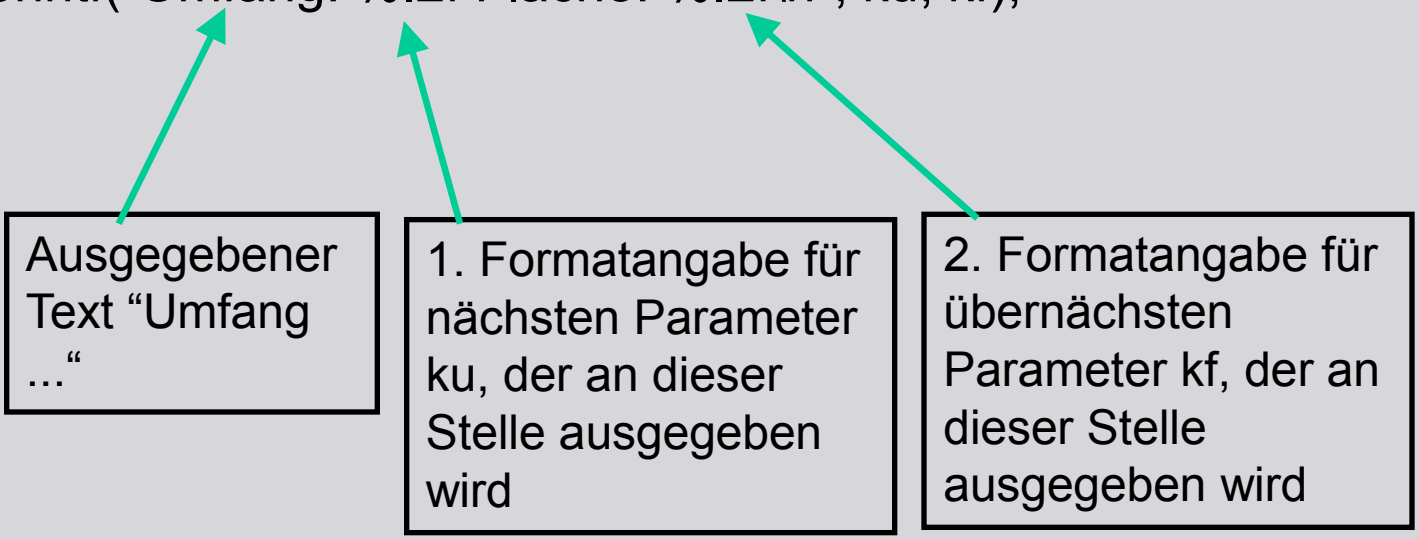
Ausgabe mehrer Werte auf Bildschirm mittels printf



Ein-/Ausgabe

Ausgabe mehrer Werte auf Bildschirm mittels printf

```
printf("Umfang: %.2f Fläche: %.2f\n", ku, kf);
```



Ein-/Ausgabe

Hochschule Deggendorf

Ausgabe mehrer Werte auf Bildschirm mittels printf

theoretisch können beliebig viele Werte verschiedener Formate mittels einer printf-Anweisung ausgegeben werden

printf(“%d%d%d%.2f%.2f ...“, 5 , 10 , 7 , 3.5 , 12.7 , ...);

in der Praxis hat jeder Compiler seine Grenzen.

Ein-/Ausgabe

Hochschule Deggendorf

Eingabe von Tastatur mittels scanf

grundsätzliche Struktur

scanf("<Format>", <Adresse>)

wobei <Format> in der Form im wesentlichen analog zu printf angegeben wird

- scanf liest in der Regel Zeichen bis zum Carriage Return
- <Adresse> gibt an, wohin im Speicher die gelesenen Daten geschrieben werden

Ein-/Ausgabe

Hochschule Deggendorf

Eingabe von Tastatur mittels scanf

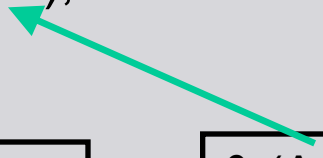
```
#include <stdio.h>

int main(void)
{
        int i;

        printf("Bitte Wert für i eingeben\n");

        scanf("%d", &i);

        printf("i hat jetzt den Wert %d\n", i);

};
```

Ein-/Ausgabe

Eingabe von Tastatur mittels scanf

```
scanf("%d", &i);
```

Formatangabe, eine Ganzzahl wird eingelesen



& (Adressoperator) gibt an, dass der eingelesene Wert an die Adresse der folgenden Variablen i geschrieben wird

Ein-/Ausgabe

Eingabe von Tastatur mittels scanf

```
int i;

scanf("%d", &i);
```

Eingabe 155 (und CR) auf Tastatur
Speichern des Werts 155 in der Variablen i

Adr. i

Arbeitspeicher
155

Ein-/Ausgabe

Hochschule Deggendorf

Eingabe mehrerer Werte von Tastatur mittels scanf

analog zu printf können mittels einer scanf-Anweisung mehrere Werte eingelesen werden

scanf(“%d%d, &i, &j);

Ein-/Ausgabe

Hochschule Deggendorf

Zum Schluss dieses Abschnitts …

Noch Fragen ??

Inhalt

Hochschule Deggendorf

- **Grundbegriffe**
  - Grundstruktur eines C Programms
  - Variable
  - Konstante
  - Typen
  - Ein-/Ausgabe
  - Operatoren
  - Kontrollstrukturen
- **Komplexe Elemente**
  - Weitere Typen
  - Pointer
  - Funktionen

Operatoren

Hochschule Deggendorf

Allgemeines zu Operatoren:

- dienen zur Verknüpfung von Daten
- dienen zur Manipulation von Daten
- können (je nach dem) auf Variable, Konstante und Ausdrücke angewendet werden

Operatoren

Hochschule Deggendorf

schon bekannt ...

- = Zuweisungsoperator weist das Ergebnis des Ausdrucks, der rechts vom Zuweisungszeichen steht an das Zuweisungsziel (meist eine Variable) zu, die links vom Zuweisungszeichen steht

- zweistellige mathematische Operatoren * (Multiplikation), / (Division), + (Addition), - (Subtraktion), verknüpfen entsprechend die Ergebnisse der zwei Ausdrücke, die links und rechts stehen miteinander

- & Adressoperator liefert die Adresse einer Variablen

Operatoren

Hochschule Deggendorf

schon bekannt ...

```
#include <stdio.h>

int main(void)
{
        int i;

        printf("Bitte Wert für i eingeben\n");

        scanf("%d", &i);

        i = i + 5 ;

        printf("i hat jetzt den Wert %d\n", i);

};
```

Operatoren

Hochschule Deggendorf

weitere (neue) ...

- % Modulo (Restbildung)
- == Vergleich (auf Gleichheit, liefert 0 zurück, falls linker Operand ungleich rechter Operand, sonst einen Wert ungleich 0)
- != Vergleich (auf Ungleichheit, liefert 0 zurück, falls linker Operand gleich rechter Operand, sonst einen Wert ungleich 0)
- <, <=, >, >= Vergleich auf kleiner, kleiner gleich, größer, größergleich, liefert Wert ungleich 0 zurück, falls Vergleich richtig, sonst 0
- &&, ||, ! logische und, logisches oder, Negation

Operatoren

Hochschule Deggendorf

weitere (neue) ...

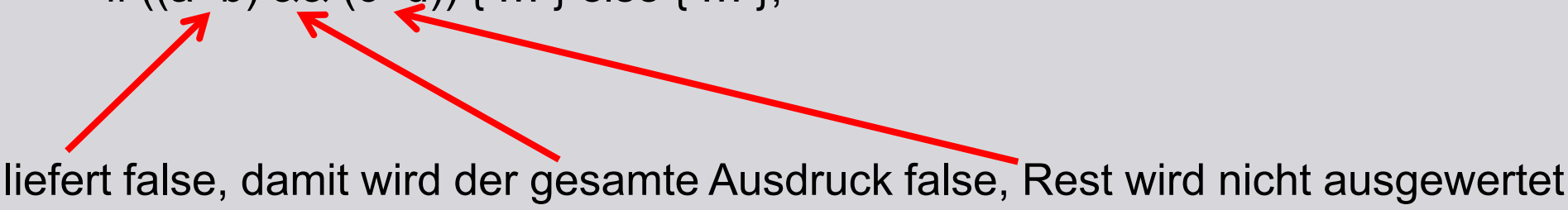
- Achtung bei && und ||:

  Die Auswertung eines Ausdrucks mit booleschen Verknüpfungen (&&, || wird beendet, sobald der Wert des Ausdrucks feststeht!

  z.B.

  int a = 3; int b = 2; int c = 5, int d = 7;

  if ((a<b) && (c<d)) { … } else { … };



Operatoren

Hochschule Deggendorf

weitere (neue) ...

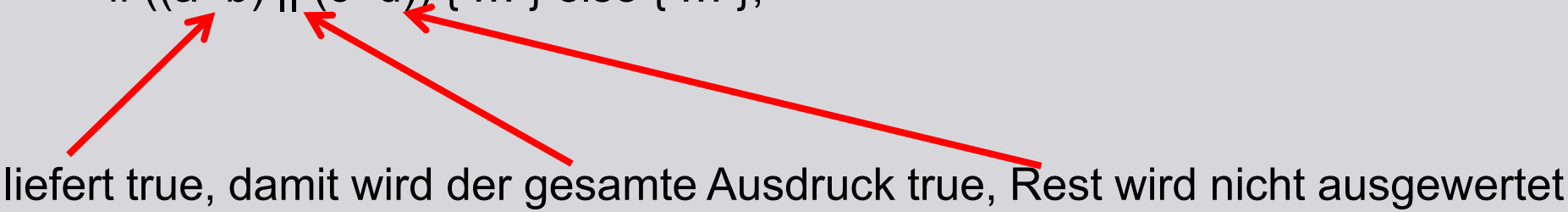
- Achtung bei && und ||:

  Die Auswertung eines Ausdrucks mit booleschen Verknüpfungen (&&, || wird beendet, sobald der Wert des Ausdrucks feststeht!

  z.B.

  int a = 3; int b = 2; int c = 5, int d = 7;

  if ((a>b) || (c<d)) { … } else { … };



Operatoren

Hochschule Deggendorf

weitere (neue) ...

- Achtung bei && und ||:

  Die Auswertung eines Ausdrucks mit booleschen Verknüpfungen (&&, || wird beendet, sobald der Wert des Ausdrucks feststeht!

  unproblematisch, so lange keine Seiteneffekte auftreten, aber:

  ```
  int a = 3; int b = 2; int c = 5, int d = 7;

  if ((a>b) || (c<d++)) { … } else { … };
  ```

  wird nicht ausgeführt

Operatoren

Hochschule Deggendorf

weitere (neue) ...

```
#include <stdio.h>

int main(void)
{
        printf("wahr oder falsch? 3 < 5 und 7 < 9: %d\n", (3<5) && (7<9));
        printf("wahr oder falsch? 3 > 5 oder 7 < 9: %d\n", (3>5) || (7<9));
        printf("wahr oder falsch? nicht 5 < 5 : %d\n", !(5<5));

};
```

Operatoren

Hochschule Deggendorf

weitere (neue) ...

```
#include <stdio.h>

int main(void)
{
        int i, j, k;

        i = 17; j = 3;

        k = i % j; /* Modulobildung */

        printf("k hat den Wert %d\n", k);
        printf("i gleich j -> Ausgabe ungleich 0: %d\n", i == j);
        printf("i ungleich j -> Ausgabe gleich 0: %d\n", i != j);
        printf("i > j -> Ausgabe                : %d\n", i > j);
        printf("i < j -> Ausgabe                : %d\n", i < j);

};
```

Operatoren

Hochschule Deggendorf

weitere (neue) ...

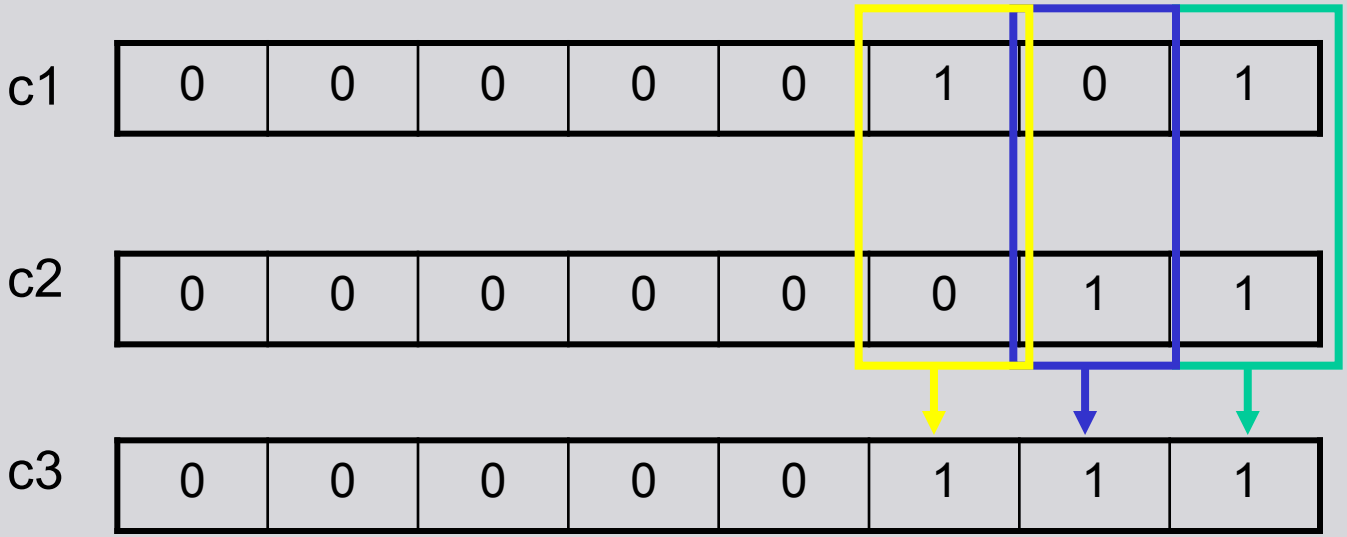
- &, |, ^, ~ bitweise Verknüpfungen *und*, *oder*, *exklusives oder*, *Negation*
- <<, >> bitweiser Linksshift, Rechtsshift
- ?: bedingter Ausdruck

Operatoren

weitere (neue) ...

- bitweise Operatoren &, |, ^, ~ verknüpfen jedes Bit des einen Operanden mit dem entsprechenden Bit des zweiten Operanden

```
unsigned char c1 = 5;
unsigned char c2 = 3;
unsigned char c3 = c1 | c2;
```



Operatoren

Hochschule Deggendorf

weitere (neue) ...

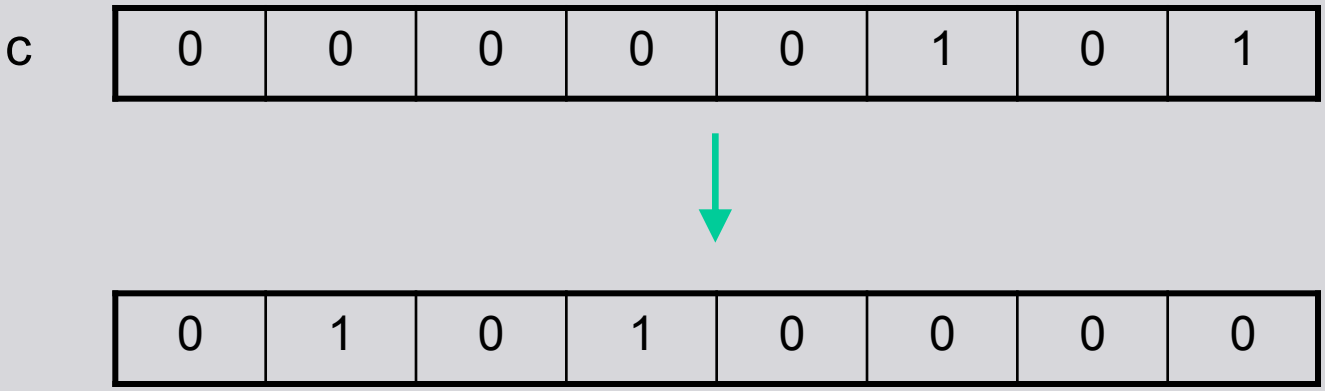
```
#include <stdio.h>

int main(void)
{
        unsigned char c1 = 5;
        unsigned char c2 = 2;
        unsigned char c3 = c1 | c2;
        printf("c1 | c2 : %d\n", c3);
        c3 = c1 & c2;
        printf("c1 & c2 : %d\n", c3);
        c3 = c1 ^ c2;
        printf("c1 ^ c2 : %d\n", c3);
};
```

Operatoren

weitere (neue) ...

- Shiftoperatoren verschieben die Bits eines Ausdrucks um die Anzahl des zweiten Operanden, z.B.

```
unsigned char c = 5;
c = c << 4; /* Verschieben um 4 Bits nach links, 0 nachziehen */
```

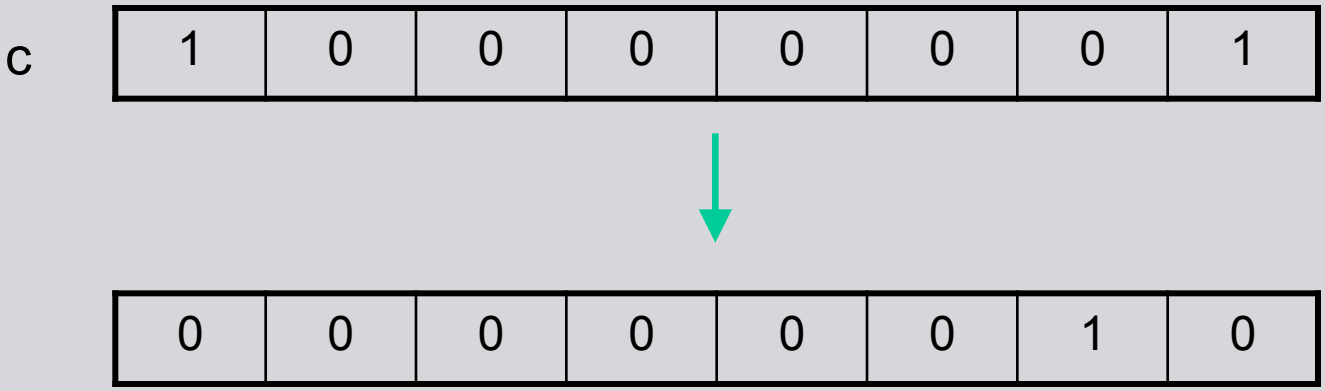


Operatoren

weitere (neue) ...

- Linksshifts ziehen immer ‘0‘ nach
- Rechtsshifts auf unsigned Größen ziehen ebenfalls ‘0‘ nach

```
unsigned char c = 129;
c = c >> 6; /* Verschieben um 6 Bits nach rechts, 0 nachziehen */
```

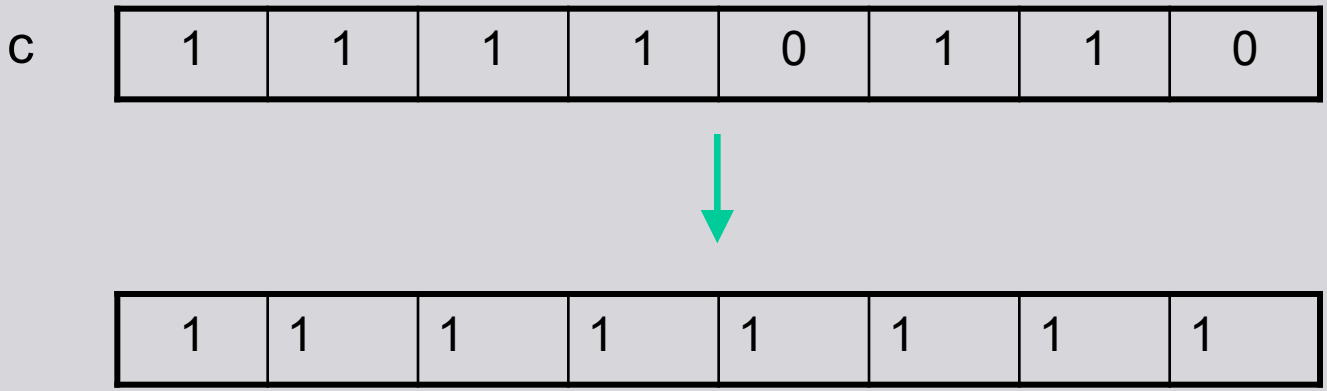


Operatoren

weitere (neue) ...

- Rechtsshifts auf signed Größen ziehen das Vorzeichenbit nach

```
char c = -10;
c = c >> 5; /* Verschieben um 5 Bits nach rechts, 1 nachziehen */
```

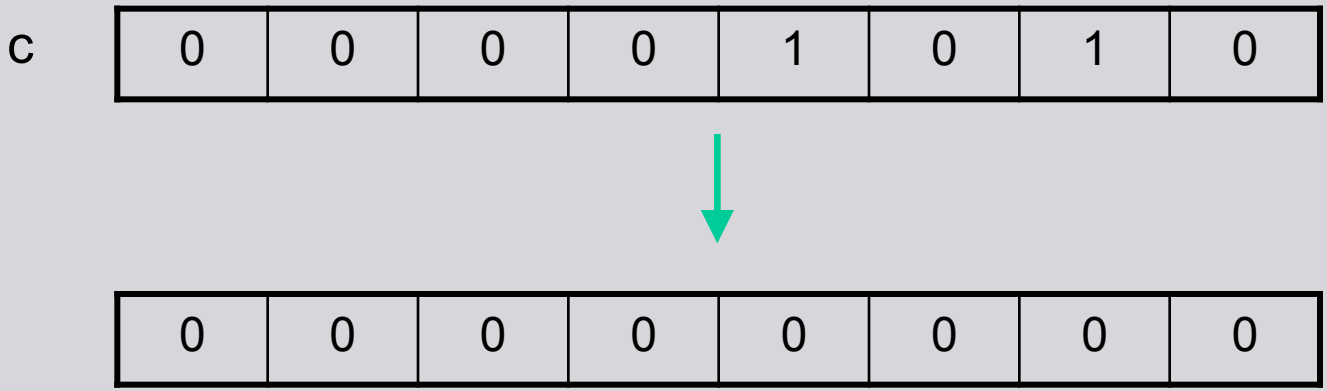


Operatoren

weitere (neue) ...

- Rechtsshifts auf signed Größen ziehen das Vorzeichenbit nach

```
char c = 10;
c = c >> 5; /* Verschieben um 5 Bits nach rechts, 0 nachziehen */
```

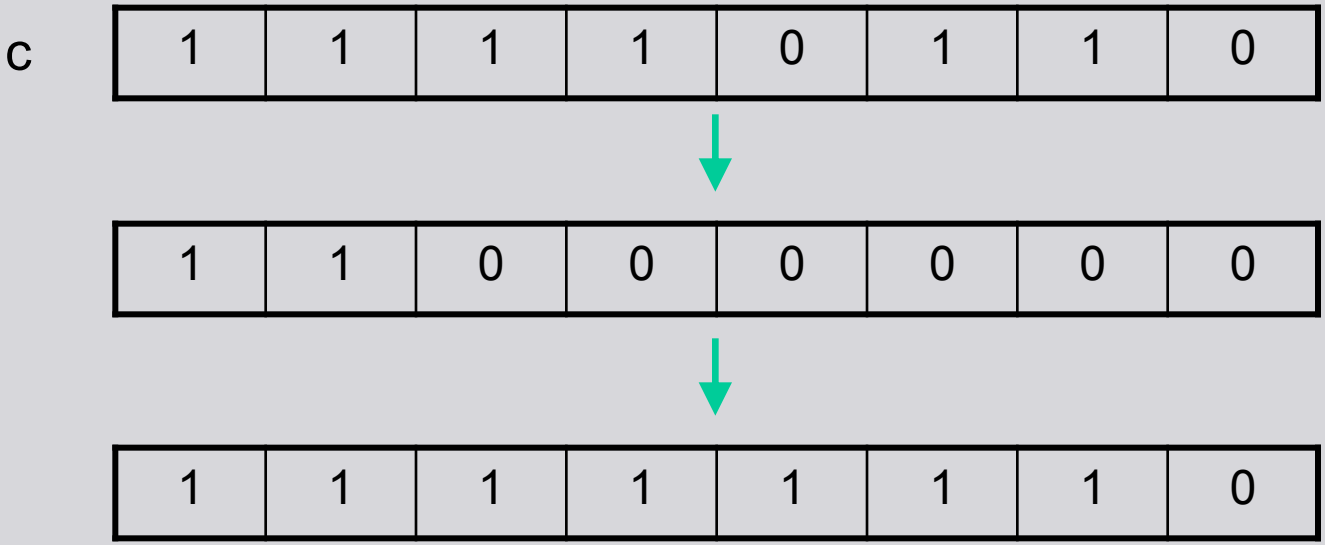


Operatoren

weitere (neue) ...

- Linksshifts auf signed Größen erhalten das Vorzeichen

```
char c = -10;
c = c << 5; /* Verschieben um 5 Bits nach links */
c = c >> 5; /* Verschieben um 5 Bits nach rechts */
```



Operatoren

Hochschule Deggendorf

weitere (neue) ...

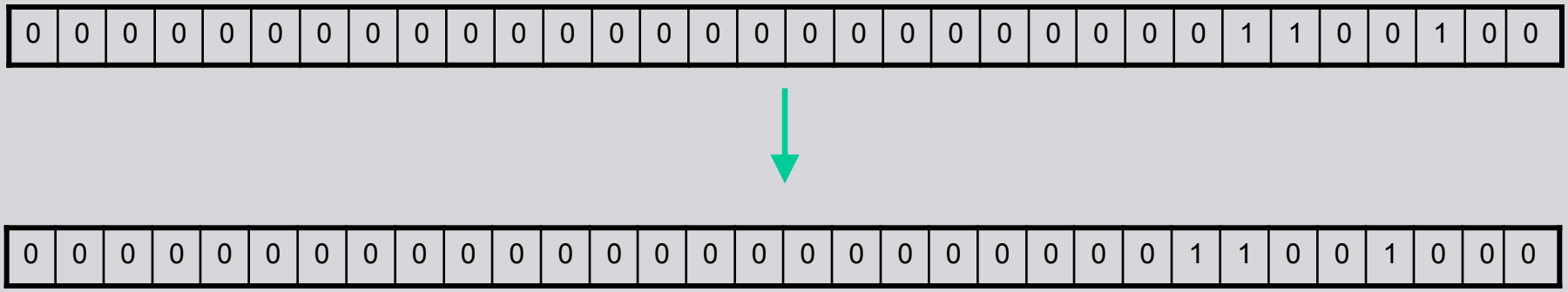
```
#include <stdio.h>

int main(void)
{
        char c = -10;
        c = c << 5; /* Verschieben um 5 Bits nach links */
        printf("%d\n", c);

        c = c >> 5; /* Verschieben um 5 Bits nach rechts */
        printf("%d\n", c);
};
```

Operatoren

Anwendungen von Shift Operatoren

- Zahlenarithmetik (+, - *, /, %, ...)
- Multiplikation mit bzw. Division durch 2er-Potenzen (Shiftoperationen)

```
unsigned int i = 100;          /* 100 = 64 + 32 + 4 = 2^7 + 2^6 + 2^2 */
i = i << 1;                    /* 2^8 + 2^7 + 2^3 = 128 + 64 + 8 = 200 */
```



Operatoren

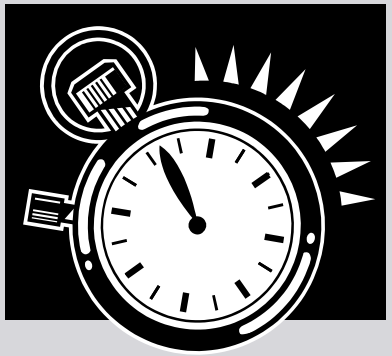
Hochschule Deggendorf

Anwendungen von Operatoren

- Multiplikation mit bzw. Division durch 2er-Potenzen mittels Shiftoperationen ist in der Regel deutlich schneller als die Anwendung des Multiplikationsoperators

  ➔ggf. Ausnutzen bei Laufzeitkritischen Anwendungen

  ➔Datenlänge beachten !



Operatoren

Hochschule Deggendorf

Anwendungen von Shift- und Bit-Operatoren

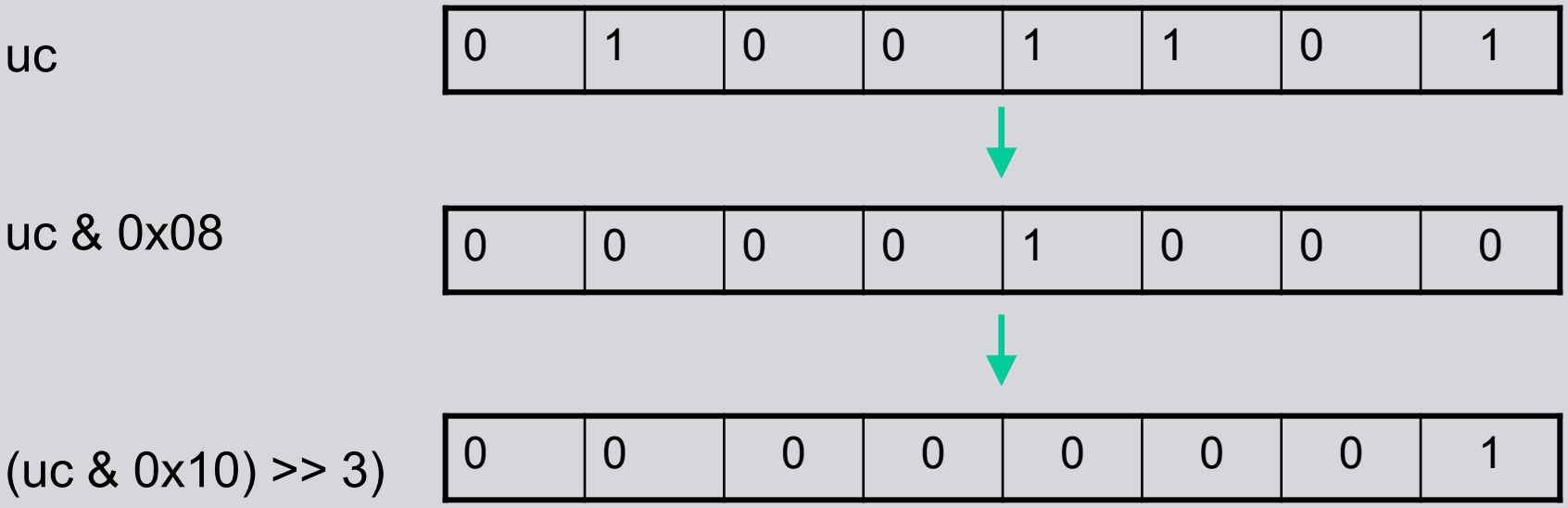
- Auswählen einzelner Datenbits durch Maskierung und Shiftoperation, z.B. Ausgabe einzelner Bits einer Variablen vom Typ unsigned char

```
unsigned char uc = 77;
printf("%d", (uc & 0x80) >> 7);        /* 1. Bit von uc */
printf("%d", (uc & 0x40) >> 6);        /* 2. Bit von uc */
printf("%d", (uc & 0x20) >> 5);        /* 3. Bit von uc */
printf("%d", (uc & 0x10) >> 4);        /* 4. Bit von uc */
printf("%d", (uc & 0x08) >> 3);        /* 5. Bit von uc */
printf("%d", (uc & 0x04) >> 2);        /* 6. Bit von uc */
printf("%d", (uc & 0x02) >> 1);        /* 7. Bit von uc */
printf("%d", (uc & 0x01) );            /* 8. Bit von uc */
```

Operatoren

Anwendungen von Shift- und Bit-Operatoren

```
unsigned char uc = 77;
...
printf("%d", (uc & 0x08) >> 3);          /* 3. Bit von uc */
...
```



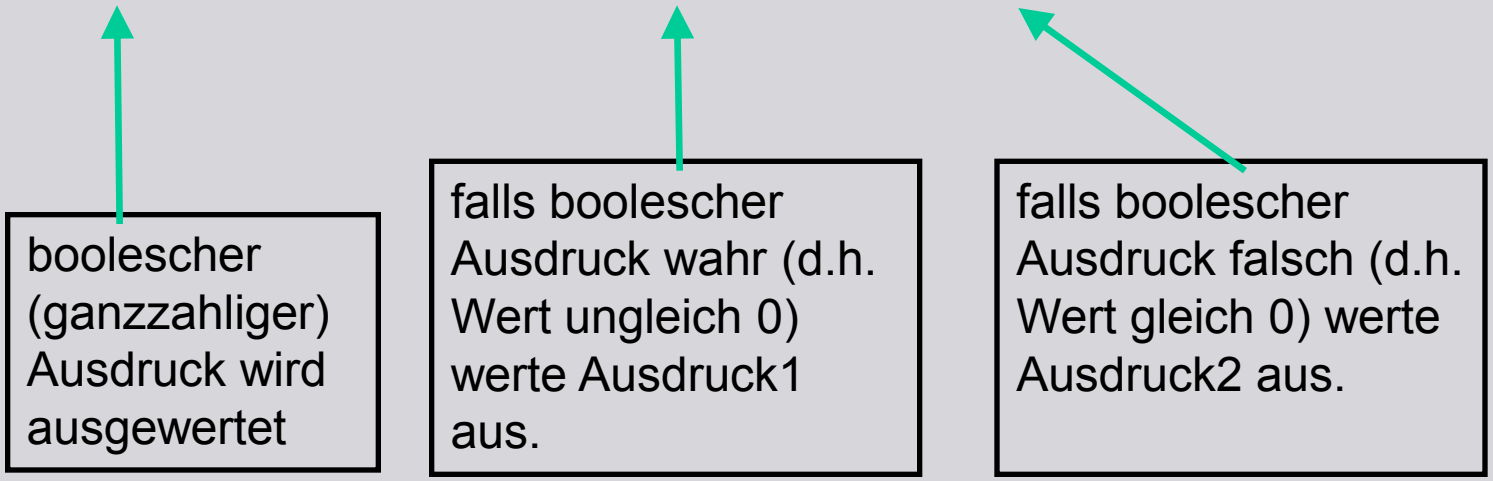
Operatoren

Hochschule Deggendorf

weitere (neue) ...

- ?: bedingter Ausdruck

boolescher Ausdruck ? Ausdruck1 : Ausdruck2



Operatoren

Hochschule Deggendorf

weitere (neue) ...

- ?: bedingter Ausdruck, Beispiel

```
int a;
int b;
int min;

printf("Bitte Wert fuer a eingeben:\n");
scanf("%d", &a);

printf("Bitte Wert fuer b eingeben:\n");
scanf("%d", &b);

min = (a<b) ? a : b;

printf("Das Minimum von %d und %d ist %d\n", a, b, min);
```

Operatoren

Hochschule Deggendorf

weitere (neue) ...

- ?: bedingter Ausdruck
  - ähnlich zu if ... else ... (s. Kap. Kontrollstrukturen)
  - schachtelbar
    (a<b) ? ((c<d) ? 1 : 2) : ((e<f) ? 3 : 4)
  - etwas unübersichtlich
  - in der Regel if ... else verwenden
  - manchmal verboten

Operatoren

Hochschule Deggendorf

weitere (neue) ...

- einstellige Operatoren
  - - Negation, z.B. int i = -(5+3);
  - ++ Inkrement, z.B. int i = 5; i++; entspricht i = i+1;
  - -- Dekrement, z.B. int i = 5; i--; entspricht i = i-1;

  Inkrement und Dekrement Operatoren können vor oder nach einer Variablen gesetzt werden, z.B. i++; ++i, --l; ++j
  Steht der Operator von der Variablen, dann wird die Operation vor der Verwendung der Variablen in einem Ausdruck durchgeführt, sonst danach.

Operatoren

Hochschule Deggendorf

weitere (neue) ...

- einstellige Operatoren
  - +=n Addition einer Zahl n zu einer anderen Zahl, z.b. x+=b; entspricht x = x+b;
  - analog -=, *=, /=, %=

Operatoren

Hochschule Deggendorf

weitere (neue) ...

- einstellige Operatoren
  - - Negation, z.B. int i = -(5+3);
  - ++ Inkrement, z.B. int i = 5; i++; entspricht i = i+1;
  - -- Dekrement, z.B. int i = 5; i--; entspricht i = i-1;

  int z = (a*b++); entspricht int z = (a*b); b++;

  int z = (a*++b); entspricht b++; int z = (a*b);

  Analog mit -- - Operator

Operatoren

Hochschule Deggendorf

Zum Schluss dieses Abschnitts …

Noch Fragen ??