

KV Betriebssysteme

Einführung in C

KV Betriebssysteme Ein- / Ausgabe

DI. Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Funktion printf



- ↪ **Zweck**
 - Formatierte Ausgabe von Werten
- ↪ **Modul**
 - #include <stdio.h>
- ↪ **Aufruf**
 - printf (Formatstring, ...);
 - Dieser Funktion können eine bel. Anzahl von Parametern übergeben werden. Der erste Parameter muß ein Formatstring sein. Dahinter folgen die auszugebenden Werte.
 - Für jeden auszugebenden Wert sollte eine Formatangabe im Formatstring vorhanden sein. Die übergebenen Werte werden nicht auf Typkorrektheit geprüft!

Formatstring von printf



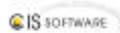
- ↪ **Syntax**
 - %[flags] [width] [.precision] [{h | l}]type
- ↪ **Flags**
 - Steuerung der Ausrichtung, Vorzeichen, Leerzeichen, ..
- ↪ **Width**
 - Breite der Ausgabe (mit *, * variable Breite)
- ↪ **Precision**
 - Gesamtanzahl der Ziffern oder Ziffern nach dem Komma
- ↪ **Type**
 - Gibt an, wie der Wert interpretiert werden soll.

Beispiel mit printf



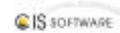
- ↪ **Beispiel**
 - #include <stdio.h>
 - int main(void) {
 printf("Name:%t%s\nAlter:%t%d", "Peter", -2);
 return 0;
}
- ↪ **Ausgabe**
 - Name: Peter
 - Alter: -2

Typ in Formatstring (I)



Typ	Datentyp	Ausgabeformat
c	int	einzelnes Zeichen
d, i	int	Dezimalzahl mit Vorzeichen
o	int	Oktalzahl ohne Vorzeichen
u	int	Dezimalzahl ohne Vorzeichen
x	int	Hexadezimalzahl ohne Vorzeichen mit den Buchstaben "abcdef"
X	int	Hexadezimalzahl ohne Vorzeichen mit den Buchstaben "ABCDEF"

Typ in Formatstring (II)

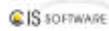


Typ	Datentyp	Ausgabeformat
e, E	double	Dezimalzahl mit Vorzeichen in Exponentialschreibweise
f	double	Dezimalzahl mit Vorzeichen in Gleitkommenschreibweise
g, G	double	e- oder f-Format
p	Pointer	Adresse in der Form xxxx:yyyy
s	String	Zeichenkette. Zeichen werden solange ausgegeben, bis das erste Nullbyte erreicht wird.

KV Betriebssysteme

Einführung in C

Beispiel mit printf (I)



```
#include <stdio.h>
int main (void) {
```

```
    char c; float f; int i;
```

```
    i = 40;
    printf ("Integer-Ausgabe:\n");
    printf ("i = %d\n",i);
    printf ("i = %5d\n",i);
    printf ("i = %-5d\n",i);
    printf ("i = %5.3d\n",i);
    printf ("i = %1d\n",i);
```

Integer-Ausgabe:

```
i = |40|
i = | 40|
i = |40 |
i = | 040|
i = |40|
```

Beispiel mit printf (II)



```
c = 'A';
printf ("Zeichen-Ausgabe:\n");
printf ("c = %c\n",c);
printf ("c = %5c\n",c);
printf ("c = %-5c\n",c);
printf ("c = %d\n",c);
printf ("c = %0c\n",c);
```

Zeichen-Ausgabe:

```
c = |A|
c = |  A|
c = |A |
c = |65|
c = |A|
```

Beispiel mit printf (III)



```
f = 839.21;
printf ("Float-Ausgabe:\n");
printf ("f = %f\n",f);
printf ("f = %10.3f\n",f);
printf ("f = %10.3e\n",f);
printf ("f = %-10g\n",f);
printf ("f = %2.1f\n",f);
return 0;
}
```

Float-Ausgabe:

```
f = |839.210022|
f = | 839.210|
f = | 8.392e+02|
f = |839.21 |
f = |839.2|
```

Zeichenweise Ein-/Ausgabe



Modul

- #include <stdio.h>

Aufruf

- Eingabe: int getchar(void);
- Ausgabe: int putchar(int c);

Beispiel

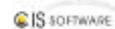
```
• char c;
c = getchar();
while (c != 0x1A) {
    putchar(c); c = getchar();
}
```

KV Betriebssysteme Operatoren

DI, Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorchestraße 2a, 4470 Enns

Operatoren



- ↳ Basisoperatoren
- ↳ Inkrement- und Dekrementoperator
- ↳ Vergleichsoperatoren
- ↳ Bitoperatoren
- ↳ Logische Operatoren
- ↳ Kurzschlußauswertung
- ↳ Auswahloperator
- ↳ Weitere Operatoren

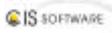
KV Betriebssysteme

Einführung in C

Basisoperatoren 

- + Addition / monadisches positives Vorzeichen
- Subtraktion / monadisches negatives Vorzeichen
- * Multiplikation
- / Division (auch für Integer!)
- % Divisionsrest
- = Zuweisung
(kann auch in Ausdrücken vorkommen)

DI. Dr. Peter René Dietmüller KV Betriebssysteme 13

Inkrement- und Dekrementoperator 

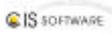
- ↪ **Inkrement**
 - ++i; i++;
- ↪ **Dekrement**
 - --i; i--;
- ↪ **Präfix / Postfix:**
 - „++i“ bedeutet Inkrement vor der Bewertung, „i++“ bedeutet Inkrement nach der Bewertung.
 - Vorsicht vor Sideeffects, wie in folgendem Beispiel:
x = 1; r = (x++ * 2) + (x+ * 3);
Welchen Wert hat x und r?

DI. Dr. Peter René Dietmüller KV Betriebssysteme 14

Vergleichsoperatoren 

Operator	Ergebnis
x < y	1 falls x kleiner als y ist, sonst 0
x <= y	1 falls x kleiner oder gleich y ist, sonst 0
x > y	1 falls x größer als y ist, sonst 0
x >= y	1 falls x größer oder gleich y ist, sonst 0
x == y	1 falls x gleich y ist, sonst 0
x != y	1 falls x ungleich y ist, sonst 0

DI. Dr. Peter René Dietmüller KV Betriebssysteme 15

Bitoperatoren 

Operator	Ergebnis
x & y	bitweises UND
x y	bitweises ODER
x ^ y	bitweises XOR
~x	Negation (Einerkomplement)
x << y	left shift von x um y Stellen (die niederwertigen Bits werden 0)
x >> y	right shift von x um y Stellen (die höherw. Bits werden bei positiven Zahlen 0, bei neg. Zahlen hängt dies vom Compiler ab)

DI. Dr. Peter René Dietmüller KV Betriebssysteme 16

Logische Operatoren 

Operator	Ergebnis
x && y	Der Ausdruck ergibt 1, wenn x und y nicht Null sind, sonst ergibt er 0.
x y	Der Ausdruck ergibt 0, wenn sowohl x als auch y Null sind, sonst ergibt er 1.
!x	Der Ausdruck ergibt 1 wenn x Null ist, sonst ergibt er 0.

DI. Dr. Peter René Dietmüller KV Betriebssysteme 17

Beispiele 

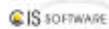
- ↪ **Bitoperatoren**
 - 2 | 193; /* 00000010 (2) | 11000001 (193) = 11000011 (195) */
 - 2 & 193; /* 00000010 (2) & 11000001 (193) = 00000000 (0) */
 - 2 << 3; /* 00000010 (2) << 3 = 00010000 (16) */
 - -16 >> 3; /* 11110000 (-16) >> 3 = 11111110 (-2) */
 - ~ 5; /* ~ 00000101 (5) = 11111010 (-6) */
- ↪ **Logische Operatoren**
 - 2 || 193; /* ergibt 1, weil beide Operanden ungleich 0 sind */
 - 2 && 193; /* ergibt 1, weil beide Operanden ungleich 0 sind */
 - ! 5 /* ergibt 0, weil der Operand ungleich 0 ist */

DI. Dr. Peter René Dietmüller KV Betriebssysteme 18

KV Betriebssysteme

Einführung in C

Kurzschlußauswertung



Erklärung

- Bei zweistelligen logischen Operatoren wird in Abhängigkeit des ersten Operanden der zweite nicht mehr ausgewertet!
- Bei `x && y` wird `y` nicht mehr ausgewertet, wenn `x` Null ist.
- Bei `x || y` wird `y` nicht mehr ausgewertet, wenn `x` ungleich Null ist.

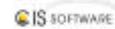
Beispiel

- `x = 1; y = (--x) && (--x);`

Anwendung

- Prüfung der Indexgrenzen: `(i <= 100) && (a[i] == 0)`

Auswahloperator



Syntax

- `x ? y : z`

Bedeutung

- Das Ergebnis des Ausdrucks hängt von `x` ab. Wenn `x` ungleich Null ist, ist das Ergebnis des Ausdrucks `y`, sonst ist es `z`.

Beispiele

- `a = 10; b = 15; c = (a > b) ? a : b;`
- `c = ,f'; c = ((c >= ,a') && (c <= ,z')) ? c - 32 : c;`

Weitere Operatoren



Operator	Name
,	Kommaoperator
()	Typkonvertierung (Cast)
&	Adressoperator
sizeof()	Größenoperator
*	Umleitungsoperator
[]	Feldindexoperator
.	Elementauswahloperator
->	Elementkennzeichnungoperator

KV Betriebssysteme

Ausdrücke

DI, Dr. Peter René Dietmüller

CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Typkonvertierung (I)



Implizite Typkonvertierung (Reihenfolge!)

Operand 1	Operand 2	Ergebnis
long double	beliebig	long double
double	beliebig	double
float	beliebig	float
unsigned long	beliebig	unsigned long
long	unsigned int	unsigned long
long	beliebig	long
unsigned int	beliebig	unsigned int
beliebig	beliebig	int

Typkonvertierung (II)



Explizite Typkonvertierung (type-cast)

- Der sogenannte cast-Operator erlaubt das explizite Umwandeln von Datentypen.

Beispiele:

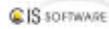
- `int i; char c;`

```
c = 'A'; /* keine Umwandlung notwendig */
i = c; /* implizite Umwandlung */
i = 'B'; /* implizite Umwandlung */
c = (char) i; /* explizite Umwandlung */
/* (wäre hier nicht notwendig!) */
```

KV Betriebssysteme

Einführung in C

Typkonvertierung (III)



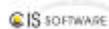
- ↪ Integer-Werte, die auf char umgewandelt werden, verlieren ihre höherwertigen Bits (genauso bei long auf int).
- ↪ Achtung: nur Konvertierungen von unsigned char in int resultieren in Werten von 0..255. Konvertierungen von char in int resultieren in Werten von -128..127.

Operatorreihenfolge (I)



- ↪ Der weiter oben in der Tabelle (folgende Seite) stehende Operator hat Vorrang.
- ↪ Bei zwei in der gleichen Zeile stehenden Operatoren entscheidet das Feld Zusammenfassung (ZF): Bei „L“ hat der linke Operator des Ausdrucks (linksassoziativ), bei „R“ der rechte (rechtsassoziativ) Vorrang.
- ↪ Ausdrücke mit logischen Operatoren (&& oder ||) werden von links nach rechts ausgewertet bis ihr Wahrheitswert klar ist (Kurzschlußauswertung)

Operatorreihenfolge (II)



P	ZF	Operator	P	ZF	Operator
01	L	() [] -> .	09	L	^
02	R	! ~ ++ -- (cast) * & sizeof	10	L	
03	L	* / %	11	L	&&
04	L	+ -	12	L	
05	L	<< >>	13	R	?:
06	L	< <= > >=	14	R	= += -= ...
07	L	== !=	15	L	,
08	L	&			

Operatorreihenfolge (III)

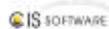


- ↪ Beispiel

• int x, y, z;	x	y	z	Ergebnis
• x = y = z = 1;	1	1	1	
• ++x ++y && ++z;	?	?	?	?
- ↪ Lösungsweg

• (++x) ((++y) && (++z))	
• 2 ((++y) && (++z))	2
• TRUE ((++y) && (++z))	2 1 1 TRUE(1)
- ↪ Anmerkung:
 - && und || Auswertung von links nach rechts!

Operatorreihenfolge (IV)



- ↪ Beispiele

int x, y, z;	x	y	z	Ergebnis
x = y = z = 1;	1	1	1	
++x && ++y ++z;	2	2	1	TRUE

int x, y, z;	x	y	z	Ergebnis
x = y = z = 1;	1	1	1	
++x && ++y && ++z;	2	2	2	TRUE

int x, y, z;	x	y	z	Ergebnis
x = y = z = -1;	-1	-1	-1	
++x && ++y ++z;	?	?	?	?

Operatorreihenfolge (V)



- ↪ Beispiele

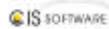
int x, y, z;	x	y	z	Ergebnis
x = y = z = -1;	-1	-1	-1	
++x ++y && ++z;	?	?	?	?

int x, y, z;	x	y	z	Ergebnis
x = y = z = -1;	-1	-1	-1	
++x && ++y && ++z;	?	?	?	?

KV Betriebssysteme

Einführung in C

Abkürzende Schreibweisen



- Die Anweisungsfolge „`int i; i = 3;`“ kann vereinfacht werden zu „`int i = 3;`“.
- Decken sich Ziel- und Quell-Operand, wie bei „`i = i + 1;`“
so kann man vereinfachend schreiben: „`i += 1;`“

- Abfragen können mit Zuweisungen verbunden werden:

```
c = getchar ();
while (c != EOF) {
    putchar (c);
    c = getchar ();
}
```

while ((c = getchar ()) != EOF)
 putchar (c);

KV Betriebssysteme

Ablaufstrukturen

DI, Dr. Peter René Dietmüller

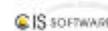
CIS-Software - Peter René Dietmüller
Lorcherstraße 2a, 4470 Enns

Ablaufstrukturen



- Anweisungen
 - Ausdrucksanweisung, z.B.: `ausdruck;`
 - Leere Anweisung, z.B.: `;`
- Block
 - Zusammenfassung mehrerer Anweisungen
 - Variablendeklaration am Beginn möglich
- Verzweigungen
 - if, switch
- Schleifen
 - for, while, do

Block



```
int main(void) {
    int a, b;
    a = 1; b = 2;
    printf("Main, a = %d, b = %d\n", a, b);
    {
        /* -- Das ist ein neuer Block -- */
        int a, c;
        a = 10; c = 20;
        printf("Block, a = %d, c = %d\n", a, c);
    }
    printf("Main, a = %d, b = %d\n", a, b);
    return 0;
}
```

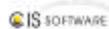
Programmausgabe

Main, a = 1, b = 2

Block, a = 10, c = 20

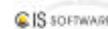
Main, a = 1, b = 2

If-Statement



- Syntax
 - if (ausdruck) ThenAnweisung else ElseAnweisung
- Bedeutung
 - Wenn der Ausdruck ungleich 0 ist, wird die „Then-Anweisung“ ausgeführt, sonst die „Else-Anweisung“.
 - Bei mehreren if-Statements gehört else immer zum letzten if. Es ist ratsam, Blöcke zu verwenden.
- Beispiel
 - if (i & 1) {
 printf ("ungerade\n"); i = i & 0xFE;
} else
 printf ("gerade\n");

Switch-Statement (I)



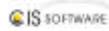
- Syntax
 - switch (Ausdruck) {
 case Konstante:; break;

 default:;
}
- Bedeutung
 - Fallunterscheidung.
 - Ab dem selektierten Zweig werden auch alle nachfolgenden Zweige abgearbeitet, bis ein break-Statement erreicht wird.

KV Betriebssysteme

Einführung in C

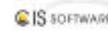
Switch-Statement (II)



↪ Beispiel

```
• switch (i) {
  case 0: printf("Null\n"); break;
  case 1: printf("Eins\n"); break;
  case 2: printf("Zwei\n"); break;
  case 3: printf("Drei\n"); break;
  case 4: printf("Vier\n"); break;
  case 5: printf("Fünf\n"); break;
  case 6: printf("Sechs\n"); break;
  default: printf("Sonst\n");
}
```

While-Schleife



↪ Syntax

- while (Bedingung) SchleifenBlock

↪ Ausführung

- Bedingung prüfen, SchleifenBlock, Bedingung ...

↪ Beispiel

```
• i = 0;
  while (i <= 255) {
    printf ("%d %c\n", i, i);
    i++;
  }
```

For-Schleife



↪ Syntax

- for (Initialisierung; Bedingung; Zähler) SchleifenBlock

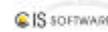
↪ Ausführung

- Initialisierung, Bedingung prüfen, SchleifenBlock, Zähler, Bedingung prüfen, SchleifenBlock, ...

↪ Beispiel

```
• for (i = 0; i <= 255; i++) {
  printf ("%d %c\n", i, i);
}
```

Do-Schleife



↪ Syntax

- do SchleifenBlock while (Bedingung);

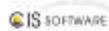
↪ Ausführung

- SchleifenBlock, Bedingung prüfen, SchleifenBlock, ...

↪ Beispiel

```
• i = 0;
  do {
    printf ("%d %c\n", i, i);
    i++;
  } while (i <= 255);
```

Break



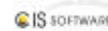
↪ Bedeutung

- Die break-Anweisung kann nur innerhalb von Schleifen oder einer switch-Anweisung verwendet werden und dient zur Beendigung der Schleife bzw. der Anweisung. break terminiert immer nur die innerste Schleife oder switch-Anweisung!

↪ Programmierstil

- Die break-Anweisung sollte nur für switch-Anweisungen und nicht für das Verlassen einer Schleife verwendet werden!

Continue



↪ Bedeutung

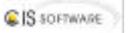
- Die continue-Anweisung kann nur innerhalb von Schleifen verwendet werden. In while- und do-Schleifen überträgt sie die Steuerung an die Testbedingung, in einer for-Schleife an den Inkrement-Ausdruck. Auch continue wirkt sich immer auf die innerste einschließende Schleife aus.

↪ Programmierstil

- continue sollte überhaupt nie verwendet werden!

KV Betriebssysteme

Einführung in C

Pythagoräische Tripel 

↳ **Problem:**

- Gesucht sind ganzzahlige Tripel (a,b,c) für die gilt:

$$a^2 + b^2 = c^2$$

↳ **Beispiel:**

- Source\Tripel.c

DI, Dr. Peter René Dietmüller KV Betriebssysteme 43
