
Zeichen

Datentyp char

`char ch = 'x'`

Zeichenvariable

Zeichenkonstante
(unter einfachen Hochkommas)

Zeichen braucht man zur Verarbeitung von Texten, Namen, Bezeichnungen.

Zeichencodes

[ASCII](#) (American Standard Code for Information Interchange)

- 1 Zeichen = 1 Byte (128 bzw. 256 Zeichen darstellbar)
- z. B. in Pascal oder C verwendet

[Unicode](http://www.unicode.org) (www.unicode.org)

- 1 Zeichen = 2 Bytes (65536 Zeichen darstellbar)
- auch Umlaute, griechische, arabische Zeichen etc.
- z. B. in Java und C# verwendet
- ASCII ist Teilmenge von Unicode

ASCII

0	NUL	16	DLE	32	space	48	0	64	@	80	P	96	'	112	p
1	SOH	17	DC1	33	!	49	1	65	A	81	Q	97	a	113	q
2	STX	18	DC2	34	"	50	2	66	B	82	R	98	b	114	r
3	ETX	19	DC3	35	#	51	3	67	C	83	S	99	c	115	s
4	EOT	20	DC4	36	\$	52	4	68	D	84	T	100	d	116	t
5	ENQ	21	NAK	37	%	53	5	69	E	85	U	101	e	117	u
6	ACK	22	SYN	38	&	54	6	70	F	86	V	102	f	118	v
7	BEL	23	ETB	39	'	55	7	71	G	87	W	103	g	119	w
8	BS	24	CAN	40	(56	8	72	H	88	X	104	h	120	x
9	HT	25	EM	41)	57	9	73	I	89	Y	105	i	121	y
10	LF	26	SUB	42	*	58	:	74	J	90	Z	106	j	122	z
11	VT	27	ESC	43	+	59	;	75	K	91	[107	k	123	{
12	FF	28	FS	44	,	60	<	76	L	92	\	108	l	124	
13	CR	29	GS	45	-	61	=	77	M	93]	109	m	125	}
14	SO	30	RS	46	.	62	>	78	N	94	^	110	n	126	~
15	SI	31	US	47	/	63	?	79	O	95	_	111	o	127	DEL

Wichtige Steuerzeichen

BS *backspace* löscht Zeichen vor Cursor
 HT *horizontal tab* Tabulatorsprung
 ESC *escape*

CR *carriage return* Zeilenvorschub
 LF *line feed* folgt auf CR
 FF *form feed* Seitenvorschub

Unicode

0000 - 007F ASCII- Zeichen
 0080 - 024F Umlaute, Akzente, Sonderzeichen
 0370 - 03FF griechische Zeichen
 0400 - 04FF cyrillische Zeichen
 0530 - 058F armenische Zeichen
 0590 - 05FF hebräische Zeichen
 0600 - 06FF arabische Zeichen

Details siehe
[http:// www. unicode. org](http://www.unicode.org)

Deutsche Umlaute

00E4 ä 00C4 Ä 00DF ß
 00F6 ö 00D6 Ö
 00FC ü 00DC Ü

Unicode

Alle Zeichen können auch mit ihrem Unicode- Wert angegeben werden:

```
'\uddd'
```

Beispiele:

```
'\u0041'    'A'  
'\u000d'    CR (carriage return)  
'\u0009'    TAB  
'\u03c0'     $\pi$ 
```

Spezielle Zeichen

```
'\n'        LF (line feed, newline)  
'\r'        CR (carriage return)  
'\t'        TAB  
'\\ '       \  
'\"'        '  
'\ddd'     Zeichenwert als Oktalzahl
```

Zeichen- Operationen

Zuweisungen:

```
char ch1, ch2 = 'a';  
ch1 = ch2;           // ok, gleicher Typ  
int i = ch2;         // ok, char kann int zugewiesen werden  
ch1 = (char) i;      // Zuweisung nach Typumwandlung möglich  
double  $\supseteq$  float  $\supseteq$  long  $\supseteq$  int  $\supseteq$  short  $\supseteq$  byte  
                     $\supseteq$  char
```

Vergleiche (==, !=, <, <=, >, >=)

Zeichen sind nach Unicode- Wert geordnet;

Buchstaben und Ziffern liegen aufeinanderfolgend

```
if ('a' <= ch && ch <= 'z' || 'A' <= ch && ch <= 'Z') ...
```

Arithmetische Operationen (+, -, *, /, %)

```
10 + (ch - '0') // Ergebnistyp: int
```

Zeichenarrays

```
char[] s = new char[ 20]; // initialisiert alle Elemente mit '\ u0000'  
char[] t = {'a', 'b', 'c'};
```

Beispiel: Textsuche

geg.: Text t , Muster pat

ges.: erstes Vorkommen von pat in t



Ergebnis: $pos \geq 0$: pat in t an Stelle pos
 $pos < 0$: pat kommt nicht in t vor

```
static int search (char[] t, char[] pat) {
    int last = t.length - pat.length;
    for (int i = 0; i <= last; i++)
        if (t[i] == pat[0]) {
            int j = 1;
            while (j < pat.length && pat[j] == t[i+j]) j++;
            // j == pat.length || pat[j] != t[i+j]
            if (j == pat.length) return i;
        }
    return -1; // not found
}
```

Beispiel: Nachbauen von readInt

```
static int readInt() {
    int val = 0;
    char ch = In.read(); // liest ein einzelnes Zeichen
    while (In.done() && '0' <= ch && ch <= '9') {
        val = 10 * val + (ch - '0');
        ch = In.read();
    } // ! In.done() || ch < '0' || ch > '9'
    return val;
}
```

Schreibtischtest: Eingabe 123+

val	ch	
0	'1' (49)	'0' = 48
1	'2' (50)	
12	'3' (51)	
123	'+' (43)	

Standardfunktionen mit Zeichen

<code>if (Character.isLetter(ch)) ...</code>	true, wenn ch ein Unicode- Buchstabe ist
<code>if (Character.isDigit(ch)) ...</code>	true, wenn ch eine Ziffer ist
<code>if (Character.isLetterOrDigit(ch)) ...</code>	
<code>if (Character.isJavaIdentifierStart(ch)) ...</code>	true, wenn ein Java- Name mit ch beginnen kann
<code>if (Character.isJavaIdentifierPart(ch)) ...</code>	true, wenn ch in einem Java- Namen vorkommen kann
<code>if (Character.isLowerCase(ch)) ...</code>	true, wenn ch ein Kleinbuchstabe ist
<code>if (Character.isUpperCase(ch)) ...</code>	true, wenn ch ein Großbuchstabe ist
<code>ch1 = Character.toUpperCase(ch2);</code>	wandelt ch2 in einen Großbuchstaben um
<code>ch1 = Character.toLowerCase(ch2);</code>	wandelt ch2 in einen Kleinbuchstaben um

Strings

Datentyp String

`String a, b;` Bibliothekstyp für 'Zeichenarrays'
`a = "Hello";` Stringkonstante (unter doppelten Hochkommas)

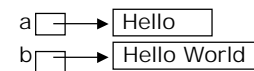
`b = a;`



```
graph LR; a[ ] --> H[Hello]; b[ ] --> H;
```

Stringvariablen sind **Zeiger** auf Stringobjekte
Stringzuweisung ist eine **Zeigerzuweisung**
Stringobjekte sind **nicht als Arrays ansprechbar**
Stringobjekte sind **nicht veränderbar**

`b = a + " World";`



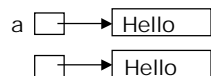
```
graph LR; a[ ] --> H[Hello]; b[ ] --> HW[Hello World];
```

Verkettung mit "+" erzeugt neues Stringobjekt
(relativ teure Operation)

Stringvergleiche

`String s = In.readName();` // liest einen Namen, z. B. Hello

`if (s == "Hello") ...` // liefert false! (Zeigervergleich)



```
graph LR; a[ ] --> H1[Hello]; b[ ] --> H2[Hello];
```

weil zwei verschiedene Objekte,
trotz gleichem Inhalt

`if (s.equals(" Hello")) ...` // liefert true! (Wertvergleich)

aber:

`String s = "Hello";`

`if (s == "Hello") ..`

// liefert true, weil gleichlautende Stringkonstanten
// nur einmal als Objekt abgespeichert werden.

trotzdem Wertvergleich immer mit equals durchführen

Stringoperationen

```
String s = "a long string";
```

```
0 1 2 3 4 5 6 7 8 9 10 11 12  
a l o n g s t r i n g
```

```
int len = s.length();
```

 liefert Anzahl der Zeichen in s
(im Gegensatz zu arr.length sind Klammern nötig!)

```
char ch = s.charAt(3);
```

 liefert das Zeichen mit Index 3 (hier 'o')

```
int i = s.indexOf("ng");
```

 liefert Index des 1. Vorkommens von "ng" in s (hier 4) oder -1

```
i = s.indexOf("ng", 5);
```

 liefert Index des 1. Vorkommens von "ng" ab Index 5 (hier 11)

```
i = s.indexOf('n');
```

 geht auch mit char

```
i = s.lastIndexOf("ng");
```

 liefert Index des letzten Vorkommens von "ng"
(Varianten wie oben)

```
String x;
```

```
x = s.substring(2);
```

 liefert Teilstring ab Index 2 (hier "long string")

```
x = s.substring(2, 6);
```

 liefert Teilstring s[2..5] (hier "long")

```
if (s.startsWith("abc")) ...
```

 liefert true, falls s mit "abc" beginnt

```
if (s.endsWith("abc")) ...
```

 liefert true, falls s mit "abc" ended

Aufbauen von Strings

aus Stringkonstante

```
String s = "very simple";
```

aus char-Array

```
char[] a = new char[10];
```

```
for (int i = 0; i < a.length; i++) a[i] = In.read();
```

```
String s1 = new String(a);
```

 // s1 enthält Kopie der Zeichen in a

```
String s2 = new String(a, 2, len);
```

 // s2 enthält Kopie von a[2..2+ len-1]

Aufbauen von Strings aus StringBuffer

aus StringBuffer (Bibliothekstyp wie String, aber modifizierbar)

```
StringBuffer b;  
b = new StringBuffer();           // erzeugt leeren StringBuffer der Länge 0  
  
len = b.length();  
b.append(x);                      // hängt x an b an. x kann beliebigen Typ haben:  
                                  // short, int, long, float, double, char, char[], String, boolean  
  
b.insert(pos, x);                 // fügt x an der Stelle pos ein (Typ von x beliebig)  
b.delete(from, to);              // löscht [from.. to] aus b  
b.replace(from, to, "abc");      // ersetzt b[ from, to] durch "abc"  
  
ch = b.charAt(i);                // wie bei String  
s = b.substring(from, to); ...  
  
b.setCharAt(pos, 'x');           // setzt b[ pos] auf 'x'  
  
s = b.toString();                // liefert Pufferinhalt als String
```

Stringkonversionen

```
int i = new Integer(" 123").intValue(); // String int  
float f = new Float(" 3.14").floatValue(); // String float
```

```
String s;  
s = String.valueOf(123);           // int String  
s = String.valueOf(3.14);         // float String
```

```
char[] a = s.toCharArray();       // String char[]  
String s = new String(a);         // char[] String
```


Beispiel: Manipulation von Datnamen

dir1/dir2/name.java name.class

- Verzeichnisse entfernen
- "java" auf "class" ändern (bzw. "class" anhängen)

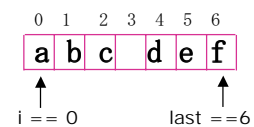
```
static string strip (String path) {  
    StringBuffer b = new StringBuffer( path); // erzeugt StringBuffer mit path als Inhalt  
    if (path.endsWith(". java") {  
        int len = path. length();  
        b.delete( len- 5, len);  
    }  
    b. append(". class");  
    int i = path.lastIndexOf( '/' );  
    if (i >= 0) b.delete( 0, i+ 1);  
    return b.toString();  
}
```

Beispiel: Wörter aus einem Text lösen

Eingabe: "Ein Text aus Woertern ..."

Ausgabe: Ein
Text
aus

```
static void printWords (String text) {  
    int i = 0, last = text. length() - 1;  
    while (i < last) {  
        //--- skip nonletters  
        while (i <= last && !Character. isLetter( text. charAt( i))) i++;  
        // end of text or text[ i] is a letter  
        //--- read word  
        int beg = i;  
        while (i <= last && Character. isLetter( text. charAt( i))) i++;  
        // end of text of text[ i] is not a letter  
        //--- print word  
        if (i > beg) System.out. println( text. substring( beg, i));  
    }  
}
```



Beispiel: Zahl in String konvertieren

Idee: Ziffern mit $n \% 10$ abspalten und in char- Array sammeln

```
static String valueOf (int n) {  
    char[] a = new char[ 20];  
    int i = 0;  
    do {  
        a[ i] = (char) (n % 10 + '0');  
        n = n / 10;  
        i++;  
    } while (n > 0);  
    StringBuffer b = new StringBuffer();  
    do {  
        i--;  
        b. append( a[ i]);  
    } while (i > 0 );  
    return b. toString();  
}
```

n	a				
154	<table border="1"><tr><td></td><td></td><td></td><td></td></tr></table>				
15	<table border="1"><tr><td>4</td><td></td><td></td><td></td></tr></table>	4			
4					
1	<table border="1"><tr><td>4</td><td>5</td><td></td><td></td></tr></table>	4	5		
4	5				
0	<table border="1"><tr><td>4</td><td>5</td><td>1</td><td></td></tr></table>	4	5	1	
4	5	1			