Datentypen in C++

Ganzzahltypen

Es gibt in C++ verschiedene Ganzzahltypen, die unterschiedliche Wertebereiche abdecken können. Hierbei ist zu beachten, dass in C++ der Wertebereich nicht exakt bestimmt ist. C++ legt **nicht exakt** fest wie viel Speicher für einen bestimmten Datentypen verwendet werden muss. Der C++ Standard sieht lediglich eine Mindestgröße vor.

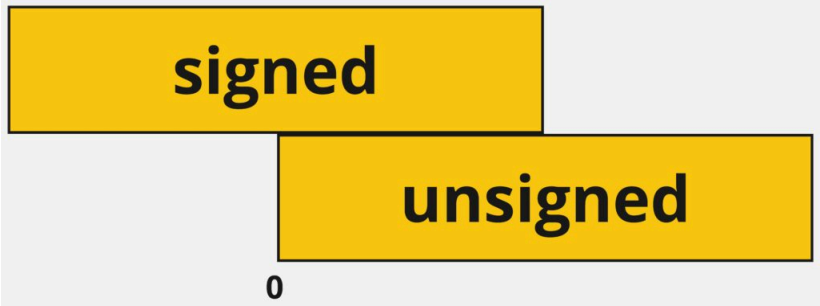
Datentyp	Bytes (mindestens)	Wertebereich (mindestens)
char	1	mit Vorzeichen (**signed**)-128 bis +127ohne Vorzeichen (**unsigned**)0 bis 255
short	2	-32.768 bis +32.767
int	2	-32.768 bis +32.767
long	4	-2.147.483.648 bis +2.147.483.647
long long	8	-9,223.372.036.854.775.808 bis +9,223.372.036.854.775.807

Mit- und ohne Vorzeichen

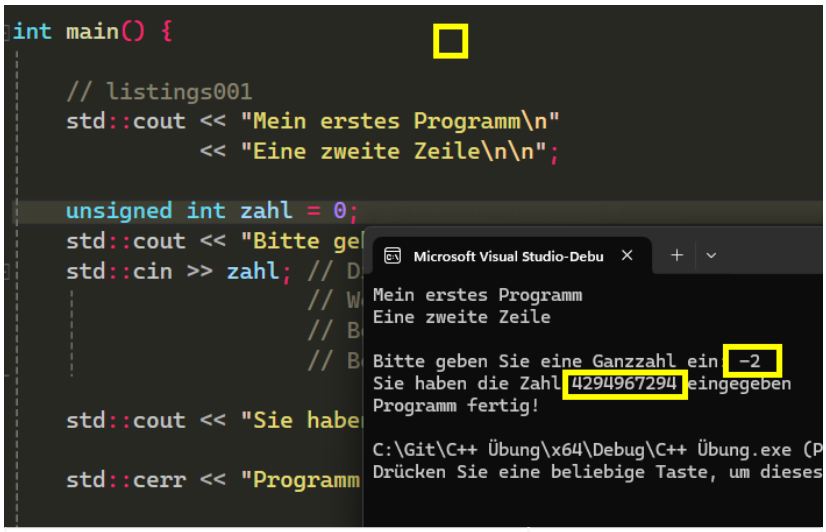
Wenn man ohne weitere Angaben einen Ganzzahltyp verwendet, steht einem per default der negative und der positive Bereich zur Verfügung. Möchte man nun allerdings festlegen, dass man die Ganzzahl ohne Vorzeichen verwenden möchte, kann das Schlüsselwort **unsigned** verwendet werden.

```
unsigned int positiv = 0;
```

Im genannten Beispiel können in der Variablen ***positiv*** nur noch positive Werte gespeichert werden. Der Wertebereich wird dadurch verschoben auf 0 bis +65.565 Ausprägungen verschoben. Die Speichermenge bleibt hierbei die gleiche!



Wenn einer vorzeichenlosen Variablen ein Wert mit Vorzeichen zugewiesen wird, wird dieser Wert in einen Wert ohne Vorzeichen konvertiert.



Dezimalzahlen	**kein Präfix**
Oktalzahlen	**0**
Hexadezimal	**0x**
Binärzahlen	**0b**

Für die bessere Lesbarkeit kann bei den Zahlen zwischen

zwei Dezimalziffern ein Apostrophzeichen eingefügt werden. Beispiel: 1’000’000

```
    /*
        Ausgabe:
        Dezimal: 254
        Oktal: 254
        Hexadezimal: 254
        Binary: 254
        Für Dezimal, Okatal, Hexadezimal
und Binärdaten
        kann der Datentyp Integer
verwendet werden.
        Der Kompiler erkennt am Präfix, ob
es sich um eine
        Dezimal-, eine Oktalzahl, einem
Hexadezimalen
        oder binären Wert handelt.
    */
    int dez = 254; // Dezimal
    int okt = 0376; // Oktal
    int hex = 0xFE; // Hexadezimal
    int bin = 0b1111'1110; // Binär
    std::cout << "Dezimal: " << dez <<
"\n";
    std::cout << "Oktal: " << okt << "\n";
    std::cout << "Hexadezimal: " << hex <<
"\n";
    std::cout << "Binary: " << bin <<
"\n";
```

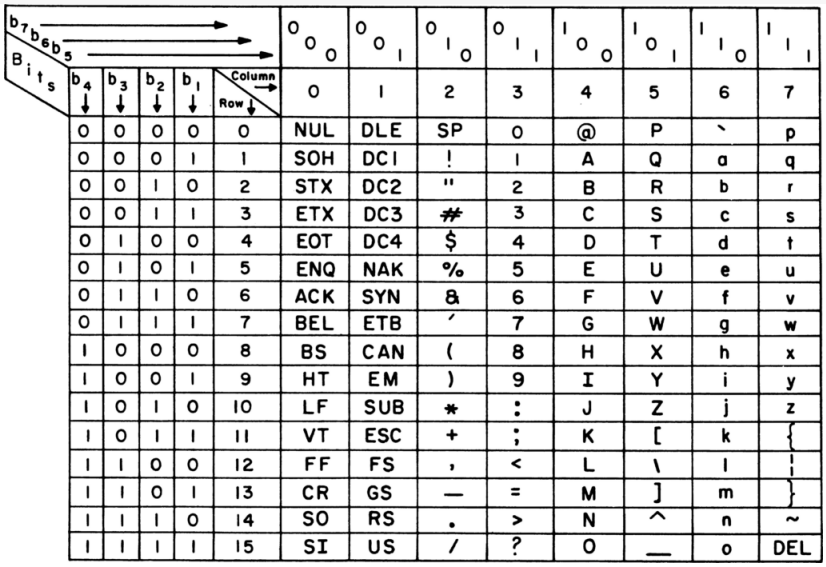
Suffixe in Ganzzahlliteralen

Einem Ganzzahlliteral kann neben einem Präfix auch ein Suffix mitgegeben werden. Hierbei kann ein Suffix bspw. kennzeichnen ob, es sich um einen Vorzeichenlosen Wert (unsigned, u oder U) oder ob es sich um einen long Wert (l oder L) handelt. Die Suffixe können hierbei auch kombiniert werden. Zum Beispiel kennzeichnet ul oder UL einen Vorzeichenlosen Long Wert.
Verzichtet man auf ein Suffix geht der Compiler immer von einem Standard-Integerwert aus, sofern dieser passt, ansonsten verwendet er den nächst größeren Typ.
Es folgt eine Tabelle mit Beispielen, die alle denselben Wert repräsentieren:

	Dezimal	Oktal	Hexadezimal	Binär (seit C++ 14)
ohne Suffix Standard-Integerwert	123	0173	0x78	0b1111011
unsigned / Vorzeichenlos	66u	0102u	0x42u	0b1000010u
Long Wert	1234567L	04553207L	0x12D687L	0b100101101011010000111L
unsigned long	9876UL	023224UL	0x2694UL	0b10011010010100UL
long long	123987LL	0362123LL	0x1E453LL	0b11110010001010011LL
unsigned long long	123987UL	0362123UL	0x1E453UL	0b11110010001010011UL

Character (char)

Der Datentyp [redacted] ist ein Ganzzahlentyp der kleinsten Art. Er ist dafür zuständig einzelne Buchstaben (Character) zu speichern. Intern kennt ein Computer nur Bitmuster, die als Binärzahlen interpretiert werden. Aus diesem Grund muss es eine Zuordnung in Form eines Codes. Ein Beispiel für solch einen Code ist der ASCII-Code:



$b_7 b_6 b_5$ → / Bits $b_4 b_3 b_2 b_1$ ↓	000	001	010	011	100	101	110	111
Column → / Row ↓	0	1	2	3	4	5	6	7
0000	0	NUL	DLE	SP	0	@	P	`	p
0001	1	SOH	DC1	!	1	A	Q	a	q
0010	2	STX	DC2	"	2	B	R	b	r
0011	3	ETX	DC3	#	3	C	S	c	s
0100	4	EOT	DC4	$	4	D	T	d	t
0101	5	ENQ	NAK	%	5	E	U	e	u
0110	6	ACK	SYN	&	6	F	V	f	v
0111	7	BEL	ETB	'	7	G	W	g	w
1000	8	BS	CAN	(	8	H	X	h	x
1001	9	HT	EM	)	9	I	Y	i	y
1010	10	LF	SUB	*	:	J	Z	j	z
1011	11	VT	ESC	+	;	K	[	k	{
1100	12	FF	FS	,	<	L	\	l	|
1101	13	CR	GS	—	=	M	]	m	}
1110	14	SO	RS	.	>	N	^	n	~
1111	15	SI	US	/	?	O	_	o	DEL

ASCII-Tabelle (Quelle: https://de.wikipedia.org/wiki/American_Standard_Code_for_Information_Interchange)

Ein anderer Code wäre der Unicode (UTF8/UTF16), die noch mehr Zeichen enthalten können. Siehe auch: https://de.wikipedia.org/wiki/Unicode).

Wenn in unserem Code also ein Zeichen wie ‘A’ angegeben wird, wird dies vom Compiler in einen numerischen Wert übersetzt. Wir können also in unserem Code auch dezimale oder Hexadezimale Zahlenwerte angeben, die den Werten der ASCII Tabelle entsprechen. Übergeben wir einer char Variablen den Wert 65, würden wir als Ausgabe ein A erhalten. Hierzu ein kleines Code-Beispiel:

```
#include <iostream>
int main()
{
    char a = 'A';
    char aHex = 0x41; //ASCII-Code von A
in Hexadezimaler Schreibweise
    char b = 'B';
```

```
    char c = 67; // ASCII-Code von C
    char d = 68;; // ASCII-Code von D
    std::cout << a << aHex << b << c << d
<< "\n";
    return EXIT_SUCCESS;
    // Ausgabe: AABCD
}
```

Nicht druckbare Zeichen

Neben einfachen Buchstaben gibt es auch sogenannte nicht-druckbare Zeichen in der ASCII-Tabelle. Das wichtigste ist hierbei – welches man auch bereits im obigen Beispiel sieht – das Newline-Zeichen (auch Linefeed oder LF genannt), welches kennzeichnet, dass eine neue Zeile begonnen wird. Solche Zeichen müssen an dieser mit einer Ersatzdarstellung, die immer mit einem Backslash eingeleitet wird, dargestellt wird. Eine neue Zeile wäre also \n. Es folgt eine Tabelle mit den wichtigsten nicht druckbaren Zeichen:

Steuerzeichen	
\n	Kennzeichnet, dass eine neue Zeile begonnen wird. Der Cursor geht dabei zum Anfang der neuen Zeile.
\r	Stellt einen sogenannten *Carriage Return* (CR) dar. Der Cursor geht dabei zum Anfang der aktuellen Zeile.
\t	*Horizontal Tab* (HT) – Stellt ein Tabulator Vorschub dar, der in der Regel vier oder acht Leerzeichen darstellt.
\”	Ermöglicht, dass das Anführungszeichen ” mit ausgegeben wird. (Beispiel: “A\”\B”” ergibt A”B”

\’	Ermöglicht, dass das Hochkommata ‘ mit ausgegeben wird.
\?	Gibt das Fragezeichen aus.
\\	Ermöglicht die Ausgabe des Backslashes in einem String.
\0	Hierbei handelt es sich um die Endmarkierung des Strings.

Es folgt ein Codebeispiel, in der einige der Steuerzeichen verwendet werden:

```
#include <iostream>
int main()
{
    char a = 'A';
    char b = 'B';
    char c = 'C';
    char d = 'D';
    std::cout << a << '\t' << b << '\n' <<
c
                  << '\t' << '\"' << d <<
'\"';
    /*
       Ausgabe:
       A       B
       C       "D"
    */
}
```

Definition und Deklaration von Variablen→