C - Deutsch Cheat Sheet

by feststelltast3 via cheatography.com/42140/cs/12707/

Basisdatentypen

Typ	Wertebereich (min.)	Formatzeichen
Ganzzahlen		
`signed char`	-128 +128	`%hhd` (für Dezimal) und `%c` (für Zeichen)
`short`	-32768 +32767	`%hd` oder `%hi`
`int`	-32768 +32767	`%d` oder `%i`
`long`	-2.147.483.648 +2.147.483.647	`%d` oder `%li`
`long long`	-9.223.372.036.854.775.808 +9.223.372.036.854.775.807	`%lld` oder `%lli`
Vorzeichenlose Ganzzahlen		
`_Bool`	0 und 1	`%u`
`unsigned char`	0 bis 255	`%hhu` (für Dezimal) `%c` (für Zeichen)
`unsigned short`	0 bis 65.535	`%hu`
`unsigened int`	0 bis 65.535	`%u`
`unsigned long`	0 bis 4.294.967.295	`%lu`
`unsigned long long`	0 bis 18.446.744.073.709.551.615	`%llu`
Fließkommazahlen		
`float`	1.2E-38 3.4E+38	`%f`
`double`	2.3E-308 1.7E+308	`%f` (`%lf` für scanf)
`long double`	3.4-4932 1.1E+4932	`%Lf`

In der Praxis empfiehlt es sich, immer den Fließkommatyp `double` zu verwenden, weil der Compiler den Typ `float` intern häufig ohnehin in den Typ `double` umwandelt.

Speicherbedarf mit sizeof ermitteln

`int ival = 0;` `sizeof(ival);`	Gibt Größe des Operanden in Byte(s) zurück.

Konstanten

Basisdatentypen (cont)

`const typ;`	Markiert die Variable als *read-only* (kann nicht mehr zur Laufzeit verändert werden).
void	
`void`	leerer Datentyp

Rechnen mit C und Operatoren

Ein- und Ausgabe benötigt den Header `<stdio.h>`

Mathematische Funktionen benötigen den Header `<math.h>`

Werte formatiert einlesen mit scanf

`scanf("%d",&ivar);-`	Formatiertes Einlesen von der Standardeingabe.

Arithmetische Operatoren

Operator	Bedeutung
`+=`	`Val1 += Val2` ist gleichwertig mit `Val1 = Val11 + Val2`
`-=`	`Val1 -= Val2` ist gleichwertig mit `Val1 = Val11 - Val2`
`*=`	`Val1 = Val2` *ist gleichwertig mit* `Val1 = Val11 Val2`
`/=`	`Val1 /= Val2` ist gleichwertig mit `Val1 = Val11 / Val2`
`%=`	`Val1 %= Val2` ist gleichwertig mit `Val1 = Val11 % Val2`

Inkrement- und Dekrement-Operator

`++`	Inkrement-Operator (Variable wird um 1 erhöht)
`--`	Dekrement-Operator (Variable wird um 1 verringert)
Anwendung	*Bedeutung*
`var++`	Erhöht den aktuellen Wert von `var`, gibt aber noch den *alten* Wert an den aktuellen Ausdruck weiter.
`++var`	Erhöht den aktuellen Wert von `var` und gibt diesen sofort an den aktuellen Ausdruck weiter.

Rechnen mit C und Operatoren (cont)

`var--`	Reduziert den Wert von `var`, gibt aber noch den alten Wert van den aktuellen Ausdruck weiter.
`--var`	Reduziert den Wert von `var` und gibt diesen sofort an den aktuellen Ausdruck weiter.

Bit-Operatoren

Bit-Operator	*Bedeutung*
`&`	bitweise UND-Verknüpfung (and)
`|`	bitweise ODER-Verknüpfung (or)
`^`	bitweises XOR
`~`	bitweise Komplement
`>>`	Rechtsverschiebung
`<<`	Linksverschiebung

Die Operanden für die Verwendung mit Bit-Operatoren müssen immer ganzzahlige Datentypen sein. `float` oder `double` dürfen nicht als Operanden verwendet werden.

Vergleichs-Operatoren

`a < b`	Kleiner als. Wahr, wenn `a` kleiner `b`.
`a <= b`	Kleiner oder gleich. Wahr, wenn `a` kleiner oder gleich groß wie `b`.
`a > b`	Größer als. Wahr, wenn `a` größer `b`.
`a >= b`	Größer oder gleich. Wahr, wenn `a` größer oder gleich groß wie `b`.
`a == b`	Gleich. Wahr, wenn `a` gleich `b`.
`a != b`	Ungleich. Wahr wenn `a` ungleich `b`.
`a && b`	Logisches UND. Wahr, wenn `a` und `b` nicht 0 sind.
`a || b`	Logisches ODER. Wahr, wenn entweder `a` oder `b` nicht 0 sind.

Implizite Typumwandlung

`(typ)a`	Gibt `a` als entsprechenden `typ` zurück.

Bedinge Anweisung und Verzweigung

if, else if, else

`if (a) {`
`  b;`
`}`
	Führt `b` aus, wenn `a` wahr ist.

`if (a) {`
`  b;`
`  c;`
`}`
	Führt `b` und `c` aus, wenn `a` wahr ist.

`if (a) {`
`  b;`
`}`
`else {`
`  c;`
`}`
	Führt `b` aus, wenn `a` wahr ist.
	Ansonsten wird `c` ausgeführt.

`if (a) {`
`  b;`
`}`
`else if (c) {`
`  d;`
`}`
`else {`
`  e;`
`}`
	Führt `b` aus, wenn `a` wahr ist,
	ansonsten `d`, wenn `c` wahr ist.
	Sollte beides unwahr sein, wird `e` ausgeführt.