# Datentypen

| | |
|---|---|
| int | Integer (int) speichert ganze Zahlen (32-bit). |
| short | short speichert ganze Zahlen (16-bit). |
| long | long speichert ganze Zahlen (32-bit oder 64-bit je nach Betriebssystem). |
| long long | long long speichert ganze Zahlen (64-bit). |
| float | float speichert Kommazahlen (32-bit). |
| double | double speichert Kommazahlen (64-bit). |
| char | Character (char) speichert einen Character (8-bit) im ASCII Format. |

# Variablen

| | |
|---|---|
| unsigned | unsigned sind Variable, welche keine negativen Werte speichern können. |
| const | const sind Variablen, welche nicht verändert werden können. |

Es ist zu beachten, dass
- keine Sonderzeichen außer „_“,
- keine Zahlen am Namensanfang,
- keine Leerzeichen,
- keine Schlüsselwörter (z.B: printf, if, while, for, …)

in den Variablennamen verwendet werden dürfen.
Spezielle Zeichen wie Umlaute können auch zu Problemen führen.
**!WICHTIG!** Variablen in C sind case sensitive

# Rechenoperatoren

| | |
|---|---|
| = | setzt eine Variable einem Wert gleich (z.B: x = 7;). |
| + | addiert zwei Werte |
| - | subtrahiert einen Wert von einem anderen. |
| * | Multipliziert zwei Wert. |
| / | Dividiert einen Wert durch den anderen (abgerundet bei Ganzzahlen) |
| % | Modulo gibt den Rest von Divisionen zurück |
| () | Man kann wie in der Mathematik Klammern für Rechnungen verwenden, sonst gilt Punkt vor Strich. |
| var++ | erhöht den Wert einer Variable um 1. |
| var-- | Verringert den Wert einer Variable um 1. |

# Logikoperatoren

Logikoperatoren geben True (Wahr) oder False (Falsch) zurück.

| | |
|---|---|
| == | vergleicht zwei Werte und gibt True zurück, wenn diese gleich sind. |
| != | vergleicht zwei Werte und gibt True zurück, wenn diese **nicht** gleich sind. |
| && | gibt True zurück, wenn beide Werte True sind. |
| \|\| | gibt True zurück, wenn mindestens ein Wert True ist. |
| ! | Invertiert den Logikwert von True zu False und False zu True. |
| < | gibt True zurück, wenn der erste Wert **kleiner** ist als der zweite. |
| > | gibt True zurück, wenn der erste Wert **größer** ist als der zweite. |
| <= | gibt True zurück, wenn der erste Wert **kleiner oder gleich** ist. |
| >= | gibt True zurück, wenn der erste Wert **größer oder gleich** ist. |

# Standard Input Output

printf()

printf gibt Text in die Konsole aus.
z.B: printf(„Hallo Welt!“);
printf kann auch Variablen ausgeben.
z.B: printf(„%d“, variable);
%d muss je nach Datentyp ersetzt werden.

| | | | |
|---|---|---|---|
| %d | int | %lf | double |
| %hd | short | %c | %char |
| %ld | long | %o | int (im Oktalsystem) |
| %lld | long long | %x | int (in Hexadezimal) |
| %f | float | %X | int (in Hexadezimal, Groß) |

Variablen können auch formatiert werden

%3d gibt mindestens 3 Stellen aus
z.B: „ 5“.
Wenn die Zahl mehr als 3 Stellen hat, werden alle Stellen ausgegeben.

%03d funktioniert wie %3d, nur dass leere Stellen mit ‚0‘ gefüllt werden
z.B: „005“.

%0.2lf gibt gerundet 2 Kommastellen aus

%4.2lf gibt mindestens 4 Ziffern inklusive gerundet 2 Kommastellen aus.

%04.2lf funktioniert wie %4.2lf nur werden leere Stellen mit ‚0‘ gefüllt.

Es gibt noch weitere Formatierung Möglichkeiten

| | |
|---|---|
| \n | neue Zeile |
| \t | Tab |
| \‘ | gibt ein ‘ aus. |
| \“ | gibt ein “ aus. |
| \\ | gibt ein \ aus. |
| %% | gibt ein % aus. |

scanf()

scanf nimmt einen Input von der Konsole und speichert den Input in eine Variable.
z.B: scanf(„%d“, &variable);
Das & muss vor dem Variablenname stehen, da die Adresse übergeben werden muss. (siehe Pointer)
%d muss je nach Datentyp ersetzt werden.

| | | | | |
|---|---|---|---|---|
| %d | int | \|\| | %f | float |
| %hd | short | \|\| | %lf | double |
| %ld | long | \|\| | %c | char |
| %lld | long long | \|\| | | |

Um chars einzulesen, muss am Ende des scanf’s %*c stehen.
z.B: scanf(„%c%*c“, &variable);
Es ist auch möglich, mehrere Variablen mit einem scanf einzulesen, indem man zum Beispiel ein Leerzeichen zwischen Variablen macht.
z.B: scanf(„%d %d“, &num1, &num2);

# Bedingungen

Bedingungen gehören zu den Kontrollstrukturen, mit ihnen kann man bestimmte Bereiche Code ausführen je nach Zustand (z.B. mit Variablen).