# Datentypen und Variablen

>> In der Informatik ist eine **Variable** ein Speicherplatz mit einem zugehörigen symbolischen **Namen**, der eine bekannte oder unbekannte Menge an Informationen enthält, die als **Wert** bezeichnet werden.

>> Im Unterschied zu einer Reihe modernerer Sprachen erfordert C, dass Sie den Datentyp jeder von Ihnen erstellten Variable bei der ersten Verwendung dieser Variable angeben.

Wir sehen uns nun die wichtigsten Datentypen an, die es in C gibt. Darüber hinaus zeigen wir zwei Datentypen, die wir Ihnen zusätzlich in Inf-Einf-B zur Verfügung stellen.

`int`

$-2^{31}$ bis $2^{31}-1$

Der Datentyp *int* (Integer) wird für Variablen verwendet, die positive oder negative Ganzzahlen enthalten.

Variablen dieses Typs belegen 4 Byte Speicher (32 Bit). Ein *int* kann also nur die Zahlen speichern, die sich mit 32 Bit darstellen lassen.

`unsigned int`

0 bis $2^{32}-1$

*unsigned* ist ein *Modifier*, der auf bestimmte Typen (etwa *int*) angewendet werden kann, wodurch der positive Wertebereich von Variablen dieses Typs effektiv verdoppelt wird, allerdings auf Kosten der Nichtzulassung von negativen Werten.

Ganzzahlen gibt es in C in 5 Größen: *char*, *short*, *int*, *long* und *long long*.

| | |
|---|---|
| char | Der Datentyp *char* wird für Variablen verwendet, die einzelne Zeichen speichern. |
| –128 bis +127 | |

Zeichen belegen in C 1 Byte Speicherplatz (8 Bits). Das bedeutet, dass der Wertebereich, den sie speichern können, auf Informationen im Wert von 8 Bits beschränkt ist.

Im ASCII-Standard wurde festgelegt, mit welchem Wert im positiven Wertebereich Zeichen wie A, B, C kodiert werden.

`float` Der Datentyp *float* wird verwendet, um reelle Zahlen („Kommazahlen“) zu speichern. Der Datentyp heißt *float*, weil er Zahlen als *floating point values* (Gleitkommawerte) speichert.

*Floats* belegen 4 Byte Speicher (32 Bit).

Der Wertebereich von *floats* ist nicht so einfach abzulesen wie bei einem *int*. Mit *floats* lassen sich größere Zahlen darstellen:

−340282346638528859811704183484516925440 bis
+340282346638528859811704183484516925440

Allerdings lassen sich nicht alle Zahlen im Wertebereich präzise speichern. Dafür können wir in *floats* auch sehr kleine Zahlen bis hinunter zu 1.4013e−45 ablegen.

`double`

Anstelle von *float* kann auch der Datentyp *double* verwendet werden.

*Doubles* benötigen 8 Byte Speicherplatz (64 Bit).

Durch die zusätzlichen 32 Bits können *doubles* Zahlen viel genauer speichern als *floats* und es lassen sich noch größere Zahlen darstellen.

Die größte darstellbare Zahl ist

179769313486231570814527423731704356798070567525844996598917476803157260780028538760589558632766878171540458953514382464234321326889464182768467546703537516986049910576551282076245490090389328944075868508455133942304583236903222948165808559332123348274797826204144723168738177180919299881250404026184124858368

`void`

*void* ist ein Typ, aber kein Datentyp, den eine Variable haben kann.

Funktionen können einen Rückgabetyp *void* haben, was bedeutet, dass sie keinen Wert zurückgeben.

Die Parameterliste einer Funktion kann auch *void* sein. Das bedeutet einfach, dass die Funktion keine Parameter entgegennimmt.

Betrachten Sie *void* vorerst wie einen Platzhalter für „nichts“. Tatsächlich ist es etwas komplexer, aber vorerst reicht diese Vereinfachung aus.

>> Das waren die fünf grundlegenden Typen, denen Sie in C begegnen werden.

In Inf-Einf-B stellen wir Ihnen außerdem zwei weitere Typen zur Verfügung, die Ihnen den Einstieg erleichtern.

`bool` Der Datentyp *bool* wird für Variablen verwendet, die einen booleschen Wert speichern sollen. Genauer gesagt, können diese Variablen nur einen von zwei Werten speichern: *true* und *false*.

Stellen Sie sicher, dass Sie

```
#include <cs50.h>
```

in Ihren Programmen verwenden, wenn Sie den Typ *bool* nutzen möchten.

`string` Der Datentyp *string* kann für Variablen verwendet werden, die Zeichenketten speichern. Zeichenketten enthalten einen oder mehrere Buchstaben, Wörter, Sätze, Absätze und dergleichen.

Stellen Sie sicher, dass Sie

```
#include <cs50.h>
```

in Ihren Programmen verwenden, wenn Sie den Typ *bool* nutzen möchten.

>> Im weiteren Verlauf des Kurses werden wir auch Strukturen (*struct*) und definierte Typen (*typedef*) kennenlernen. Diese geben Ihnen mehr Flexibilität, um neue Datentypen zu erstellen, die Sie in manchen Programmen benötigen.

Nun sehen wir uns noch an, wie man Variablen erzeugt und manipuliert.

## Anlegen einer Variable

Um eine Variable anzulegen, müssen Sie lediglich den Datentyp der Variablen angeben und ihr einen Namen geben:

```
int number;  // Deklaration
char letter; // declaration
```

Wenn Sie mehrere Variablen desselben Typs erstellen möchten, geben Sie den Namen des Typs einmal an und listen dann so viele Variablen dieses Typs auf, wie Sie möchten:

```
int height, width;
float sqrt2, sqrt3, pi;
```

Es ist eine gute Praxis, Variablen erst an der Stelle zu deklarieren, ab der man sie braucht.

## Verwenden von Variablen

Nachdem eine Variable deklariert wurde, wird der Typ nicht mehr angegeben.

```
int number;   // Deklaration
number = 17;  // Zuweisung
char letter;  // declaration
letter = 'H'; // assignment
```

Sie können eine Variable gleichzeitig deklarieren und ihr einen Wert zuweisen. Dies wird auch als *Initialisieren* bezeichnet.

```
int number = 17;  // Initialisierung
char letter = 'H'; // initialization
```

>> *Damit schließen wir die Behandlung von Datentypen und Variablen ab.*

>> *Bis bald!*

# EXTRAS IN 3 MINUTEN

FRAGEN – ANTWORTEN – RÄTSEL
UND KURZE ZUSAMMENFASSUNG

F1: Warum müssen wir unterschiedliche Größen für Ganzzahlen und Gleitkommazahlen verwenden? Warum sollte ich *float* nehmen, wenn es auch *double* gibt?

A2: Ein *char* ist im Grunde eine 8-Bit-Ganzzahl, die Werte von −128 bis 127 speichern kann.

Die positiven Zahlen sind gemäß der ASCII-Kodierung mit Zeichen verbunden. So steht beispielsweise die Zahl 65 für das Zeichen „A“, 97 für „a“ usw.

Wenn Sie einer *char*-Variablen ein Zeichen zuweisen, verwenden Sie einfache Anführungszeichen:

```
char letter = 'A';
```

Der Computer speichert in *letter* dann die Zahl 65, aber wenn Sie die Variable als *char* ausgeben lassen, sehen Sie ein „A“.

F3: Was passiert, wenn ich einem *unsigned int* eine negative Zahl zuweise?

F4: Was passiert, wenn ich einen *float* einem *int* zuweise?

F5: Was passiert, wenn ich versuche, eine Zahl zu speichern, die zu groß für einen *int* ist?“

F6: Was passiert, wenn ich eine *char*-Variable verwende, um eine Zahl außerhalb des ASCII-Bereichs zu speichern?

## Rätsel-hafter Fehler

Nehmen wir an, dein Programm so aus:

```c
#include <stdio.h>
void main() {
    float result = 3 / 4;
    printf("3 / 4 = %f\n", result);
    // %f wird durch den Wert von
    // result ersetzt und als
    // Kommazahl ausgegeben
}
```

Warum rechnet es falsch?

Um das zu verstehen, muss man wissen, wie C mit Literalen umgeht.

## Literale in C

Literale sind *konstante Werte*, die direkt in den Quellcode geschrieben werden:

Ganzzahlige Literale: 42, 0, –17

Fließkomma-Literale: 3.14, –0.01, 2.0e–3

Zeichenliterale: 'A', '7', '\n'

Stringliterale: "Hallo, Welt!", ""

Ganzzahlige Literale werden standardmäßig als *int* behandelt. Wenn alle Operanden Ganzzahlen sind, führt C bei der Durchführung von Operationen Ganzzahlarithmetik aus.

# ZUSAMMENFASSUNG

Variablen haben einen Datentyp, der explizit angegeben werden muss.

Man sagt auch:
C ist „statisch typisiert“.

Variablen erst bei Bedarf deklarieren, uninitialisierte Variablen nicht auslesen.

Wichtige Typen sind
int, char, float und double.

Inf-Einf-B fügt hinzu:
bool und string

Typen haben unterschiedliche Zahlenbereiche und Genauigkeit.

Auch Literale sind typisiert:
Ganzzahlen (42),
Gleitkomma-Zahlen (3.14),
Zeichen ('A'), String ("Hallo").

Achtung: Ganzzahl-Literale werden als *int* behandelt.

# AUFLÖSUNG ZU F3–F6

F3: Der Compiler könnte Sie warnen, aber es wird trotzdem kompiliert. Die negative Zahl wird in eine große positive Zahl umgewandelt. Dies geschieht, weil die binäre Darstellung der negativen Zahl als positive Zahl interpretiert wird. Wenn Sie z. B. -1 einer vorzeichenlosen int-Zahl zuweisen, erhalten Sie den maximalen Wert, den die vorzeichenlose int-Zahl enthalten kann (normalerweise 4.294.967.295).

F4: Der Dezimalteil wird abgeschnitten (nicht gerundet). Beispiel: int x = 3.7; der Wert von x ist dann 3. Dies kann zu einem Verlust an Genauigkeit führen, seien Sie also vorsichtig. Der Compiler gibt möglicherweise eine Warnung zu dieser Umwandlung aus.

F5: Dies wird als Integer-Überlauf (integer overflow) bezeichnet. Das Verhalten ist in C undefiniert, was bedeutet, dass es zu unvorhersehbaren Ergebnissen führen kann. Normalerweise wird die Zahl zu einer negativen Zahl oder einer kleineren positiven Zahl ("wrap around"). Wenn Sie z. B. 1 zur größtmöglichen int-Zahl addieren, erhalten Sie möglicherweise die kleinstmögliche int-Zahl.

F6: Werte von -128 bis -1 werden gespeichert, entsprechen jedoch nicht den Standard-ASCII-Zeichen. Wenn diese als Zeichen gedruckt werden, erscheinen Sonderzeichen. Welches Zeichen gedruckt wird hängt von der konfigurierten Zeichencodierung des Systems ab. Beispiel: char x = 200; könnte beispielsweise als „È" ausgegeben werden.