

Zahlen von Systemen aussprechen lassen



... noch **zwei-sieben-drei**
Kilometer bis zum Ziel

Darlehenssumme

Gesamtbetrag des Darlehens

14892,20 Euro

Betrag in Worten

vierzehntausendachthundertzweiundneunzig-komma-zwanzig

Was erwartet mich?

- Aufbau des Dezimalsystem
- Entstehung und Wissenswertes
- Einflüsse anderer Zahlensysteme

- Analyse der Aussprache von 1er, 10er und 100er Stellen
- Algorithmus zum Aussprechen von 100er-Blöcken

- Aussprache höherer Stellen
- Algorithmus zum Aussprechen größerer Zahlen

- Wie funktioniert die Sprachausgabe?

- Ein wenig unnützes Wissen 😊

Das weltweit am meisten verwendete Zahlensystem ist das Dezimalsystem. Auch wir Europäer verwenden es.

Stellen wir uns zunächst die Frage:

Wie ist das Dezimalsystem aufgebaut?

Das Dezimalsystem

10^{12}	10^9	10^6	10^3	10^2	10^1	10^0
Billion	Milliarde	Million	Tausend	Hundert	Zehn	Eins
1 000 000 000 000	1 000 000 000	1 000 000	1 000	100	10	1

Das Dezimalsystem wird auch **Denärsystem** oder **Zehnersystem** (*lat. Decimus = der Zehnte*) genannt. Es ist ein **Stellenwertsystem mit der Basis 10**. Jede Zahl lässt sich daher als Summe von Potenzen der Basis 10 darstellen:

Beispiel: 4827	7	*	10^0	=	7
	2	*	10^1	=	20
	8	*	10^2	=	800
	4	*	10^3	=	4000
	<hr/>				
	Summe		=	4827	

Wissenswertes zu Zahlensystemen

- Das Dezimalsystem stammt ursprünglich aus Indien und entstand vermutlich aus der Tatsache, dass der Mensch **zehn Finger** hat, welche er als Rechenhilfe und zum Zählen verwenden konnte.
- Ein aus mathematischen Gesichtspunkten besseres System wäre das **Duodezimalsystem zur Basis 12** gewesen, da 12 mehrere Teiler (2, 3, 4 und 6) bietet.
- Die Babylonier und Sumerer verwendeten vermutlich aufgrund ihrer hochentwickelten Astronomie und der Einteilung des Jahres in 360 Tage ein 60er-System, welches zusätzlich eine restfreie Teilung durch 5 ermöglichte ($60/5=12$).
- Auch auf 20 basierende Systeme waren im alteuropäischen Raum verbreitet.

Bis heute ist das Dutzend (=12) eine feste Handelseinheit und auch der Tag wird in 2 mal 12 Stunden á 60 Minuten unterteilt...
Ebenso verwenden wir für Kreise die 360° Einteilung ($6 \cdot 60^\circ$)

Wie werden nun deutsche Zahlennamen gebildet?

Wir untersuchen das Dezimalsystem

Untersuchung des Bereichs 0 bis 20

10 ⁰	Ableitung: 10 bis 20
Null	Einzig Zehn
Ein	Einzehn Elf
Zwei	Zweizehn Zwölf
Drei	Dreizehn
Vier	Vierzehn
Fünf	Fünfzehn
Sechs	Sechszehn Sechzehn
Sieben	Siebenzehn Siebzehn
Acht	Achtzehn
Neun	Neunzehn

Elf und Zwölf stehen als eigenständige Namen ohne die Nachsilbe „-zehn“.

[im Gotischen: *anlif* und *twalif* mit der Nachsilbe „-lif“ (=„das darüber hinaus gehende“)]

Die typische Und-Verknüpfung von Einer- und Zehnerstelle (z. B. fünf-UND-sechzig) entfällt bei den Zahlen von 13 bis 19.

Diese Unregelmäßigkeiten im unteren Zahlenbereich stammen aus früheren Ansätzen für auf 12 bzw. 20 Zahlen basierte Systeme.

Untersuchung der 10er-Stellen

10^0	10^1 (Zehnerstellen)
Ein	Einzig Zehn
Zwei	Zweizig Zwanzig
Drei	Dreizig Dreißig
Vier	Vierzig
Fünf	Fünzig
Sechs	Sechszig Sechzig
Sieben	Siebenzig Siebzig
Acht	Achtzig
Neun	Neunzig

Leider gibt es auch in den Zehnerstellen viele Ausnahmen, so dass die Regel:
Einerstelle + „zig“ nicht allgemeingültig ist! ☹

Verknüpfung mit Einerstelle:

Im Deutschen wird die Einer-Einheit zuerst genannt und mit „UND“ mit der Zehnereinheit verbunden.

Beispiel: *drei-UND-zwanzig*
(Vergl. Englisch: *twenty-three*)

Untersuchung der 100er-Stellen

10^0	10^2 (Hunderterstellen)
Ein	Ein-hundert
Zwei	Zwei-hundert
Drei	Drei-hundert
Vier	Vier-hundert
Fünf	Fünf-hundert
Sechs	Sechs-hundert
Sieben	Sieben-hundert
Acht	Acht-hundert
Neun	Neun-hundert

Für die Bildung der 100er-Stellen kann nun endlich eine Regel erfasst werden:

[Zahlenname an Position 10^2] + „hundert“

Handelt es sich um eine ungerade 100er-Stelle, wird der Rest einfach hinten angehängt:

438 = [Vier-hundert] + [acht-und-dreißig]

905 = [Neun-hundert] + [fünf]

Ausnahme: Endet die 100er-Stelle auf „01“ muss am Ende zusätzlich ein „s“ angehängt werden!

301 = [Drei-hundert] + [ein] + [s]

Analyse großer Zahlen

Unabhängig davon, wie groß eine Zahl ist, werden 100er-Blöcke immer gleich ausgesprochen. Es wird lediglich der Name des Stellenwerts (z. B. –Million[en], -Tausend, etc.) angehängt.

Beispiel: 18 342 576 918 864

Achtzehn Billionen

Dreihundertzweiundvierzig Milliarden

Fünfhundertsechundsiebzig Millionen

Neunhundertachtzehn Tausend

Achthundertvierundsechzig

Daher ist es naheliegend, die Zahl zunächst in Dreier-Blöcke (=100er-Blöcke) zu zerlegen und sich mit deren Aussprache zu befassen!

Zerlegung in 100er-Blöcken

Die Zahl wird einfach durch Modulo-Division mit 1000 zerlegt, und der Rest in ein Array geschrieben (*siehe Code-Beispiel in „C“*). So steht die niedrigste Stelle auch gleich am niedrigsten Index.

```
int i = -1;
while (zahl > 0)
{
    i=i+1;
    DreierGruppe[i] = zahl % 1000;
    zahl = zahl / 1000;
}
```

Beispiel: 342 576 918 864

Index	Wert
0	864
1	918
2	576
3	342

Listen mit Zahlennamen festlegen

Die Voruntersuchung des Zahlensystems zeigte, dass es sinnvoll bis notwendig ist, dem System Listen mit den Zahlennamen mitzuteilen.

Aufgrund der vielen Unregelmäßigkeiten im unteren Zahlenbereich, werden die Zahlen 0 bis 19 fest mitgegeben (*Liste „kleiner20“*).

Auch die Zehnerstellen wären nur mit vielen Korrekturen automatisch zu generieren. Deshalb ist es sinnvoller, auch diese zu hinterlegen (*Liste „zehner“*).

Liste „kleiner20“	
0	null
1	ein
2	zwei
3	drei
4	vier
5	fünf
6	sechs
7	sieben
8	acht
9	neun
10	zehn
11	elf
12	zwölf
13	dreizehn
14	vierzehn
15	fünfzehn
16	sechzehn
17	siebzehn
18	achtzehn
19	neunzehn

Liste „zehner“	
0	null
1	zehn
2	zwanzig
3	dreißig
4	vierzig
5	fünfzig
6	sechzig
7	siebzig
8	achtzig
9	neunzig

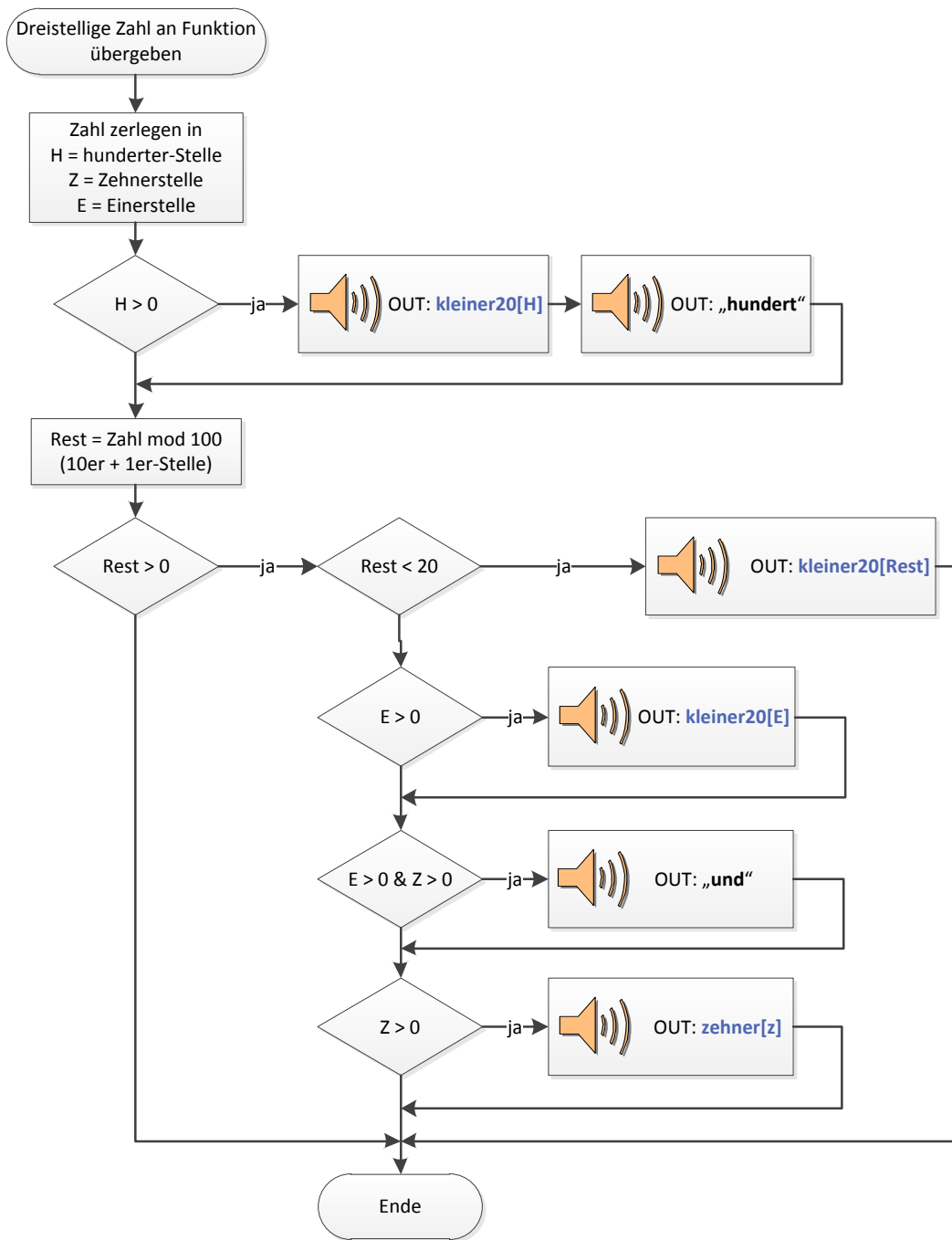
Code-Beispiel in C

Listen mit Zahlennamen festlegen

```
static char kleiner20[][10] =
{
    {"\0"},
    {"ein\0"},
    {"zwei\0"},
    {"drei\0"},
    {"vier\0"},
    {"fünf\0"},
    {"sechs\0"},
    {"sieben\0"},
    {"acht\0"},
    {"neun\0"},
    {"zehn\0"},
    {"elf\0"},
    {"zwölf\0"},
    {"dreizehn\0"},
    {"vierzehn\0"},
    {"fünfzehn\0"},
    {"sechzehn\0"},
    {"siebzehn\0"},
    {"achtzehn\0"},
    {"neunzehn\0"},
};
```

```
static char zehner[][10] =
{
    {"\0"},
    {"\0"},
    {"zwanzig\0"},
    {"dreißig\0"},
    {"vierzig\0"},
    {"fünfzig\0"},
    {"sechzig\0"},
    {"siebzig\0"},
    {"achtzig\0"},
    {"neunzig\0"},
};
```

100er-Block ausgeben



Liste „kleiner20“

0	null
1	ein
2	zwei
3	drei
4	vier
5	fünf
6	sechs
7	sieben
8	acht
9	neun
10	zehn
11	elf
12	zwölf
13	dreizehn
14	vierzehn
15	fünfzehn
16	sechzehn
17	siebzehn
18	achtzehn
19	neunzehn

Liste „zehner“

0	null
1	zehn
2	zwanzig
3	dreißig
4	vierzig
5	fünfzig
6	sechzig
7	siebzig
8	achtzig
9	neunzig

Code-Beispiel in C

100er-Block ausgeben

```
void SprichDreier(int zahl)
{
    int h = 0; // Hunderter-Stelle
    int z = 0; // Zehner-Stelle
    int e = 0; // Einer-Stelle

    h = zahl / 100;      // Zahl in 100er, 10er und 1er-Stelle zerlegen
    zahl = zahl % 100;

    z = zahl / 10;       // In Zahl steht 10er + 1er-Stelle (Rest von Modulo 100), in z nur die Zehnerstelle
    e = zahl % 10;

    if (h > 0) { printf("%s hundert ", kleiner20[h]); } // Hunderter-Stelle ausgeben

    if (zahl > 0) // Wenn noch 10er oder 1er-Stelle vorhanden ist
    {
        if (zahl < 20)
        {
            printf("%s", kleiner20[zahl]); // Zahlennamen < 20 direkt aus Array holen
        }

        else
        {
            if (e > 0) { printf("%s", kleiner20[e]); } // 1er-Stelle ausgeben, wenn vorhanden
            if ((e > 0) && (z > 0)) { printf(" und "); } // 1er und 10er-Stelle mit "UND" verbinden
            if (z > 0) { printf("%s", zehner[z]); } // 10er-Stelle ausgeben, wenn vorhanden
        }
    }
}
```

Korrektur der Ausgabe

Durch den Algorithmus können wir nun dreistellige Zahlen aussprechen lassen. Jedoch treten in einigen Fällen noch kleine **Ungenauigkeiten** auf:

Zahl	Ausgabe	Ausnahme
1	ein	<i>100-Block = 1</i>
901	neun-hundert-ein	<i>100-Block endet auf „01“</i>

Untersuchen wir nun höhere Stellen (ab 10^3), so werden wir feststellen, dass auch hier Sonderfälle zu beachten sind. Daher untersuchen wir nun zunächst die größeren Zahlennamen und korrigieren beim Verbinden der einzelnen 100er-Blöcke.

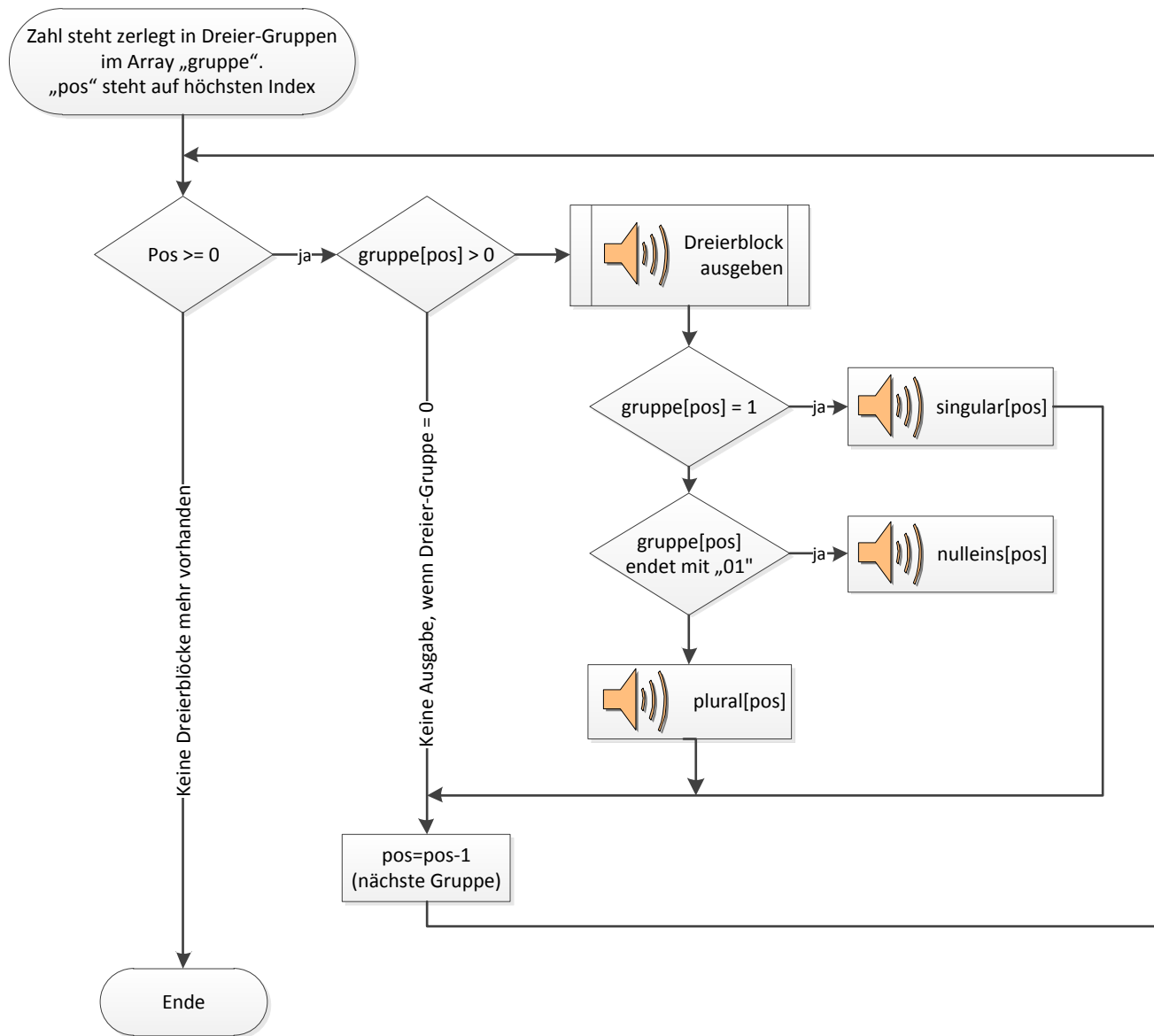
Untersuchung höherer Stellen

10^0	10^3 (Tausend)	10^6 (Million)	10^9 (Milliarde)
Ein	Ein-tausend	Ein + [e] Million	Ein + [e] Milliarde
Zwei	Zwei-tausend	Zwei Million + [en]	Zwei Milliarde + [n]
Drei	Drei-tausend	Drei Million + [en]	Drei Milliarde + [n]
Vier	Vier-tausend	Vier Million + [en]	Vier Milliarde + [n]
Fünf	Fünf-tausend	Fünf Million + [en]	Fünf Milliarde + [n]
Sechs	Sechs-tausend	Sechs Million + [en]	Sechs Milliarde + [n]
Sieben	Sieben-tausend	Sieben Million + [en]	Sieben Milliarde + [n]
Acht	Acht-tausend	Acht Million + [en]	Acht Milliarde + [n]
Neun	Neun-tausend	Neun Million + [en]	Neun Milliarde + [n]

Ab einer Million müssen wir zusätzlich unterscheiden, ob es sich um **Singular** (Eine Million) oder **Plural** (Zwei Millionen) handelt und dementsprechend korrigieren!
Auch Dreierblöcke die auf **x01** bei $\{x|x>0\}$ enden unterliegen einer Ausnahme:

Beispiel: Ein[e] Million – Zwei Million[en] – Vierhundert[s] Million[en]

Dreierblöcke verbinden und korrigieren



Liste „singular“ (001)

0	s
1	tausend
2	e Million
3	e Milliarden

Liste „nulleins“ (x01)

0	s
1	stausend
2	s Millionen
3	s Milliarden

Liste „plural“ (ab 2)

0	
1	tausend
2	_Millionen
3	_Milliarden

Code-Beispiel in C

Dreierblöcke verbinden und korrigieren

```
while (index >= 0)
{
    if (DreierGruppe[index] > 0)    // Wenn der Wert der Gruppe nicht 0 ist, dann ausgeben
    {

        SprichDreier(DreierGruppe[index]);

        // Korrekturen anhängen
        if ( DreierGruppe[index] == 1 )           // bei 001
        {
            printf("%s", singular[index]);
        }
        else if ( DreierGruppe[index]%100 == 1) // bei x01
        {
            printf("%s", nulleins[index]);
        }
        else
        {
            printf("%s", plural[index]);
        }
    }

    index--;
}
```

Audio-Ausgabe

Bisher wird die Zahl nur am Bildschirm ausgegeben. Möchten wir die Zahl vom System auch aussprechen lassen, müssen wir für jeden Zahlennamen aus den verwendeten Listen („kleiner20“, „zehner“, „singular“, usw...) ein Audiofile mit dem entsprechenden Text erstellen.

Anstatt den Zahlennamen nun mit auszugeben, spielen wir einfach an der Stelle das entsprechende Audio-File ab.

```
// Für C/C++, Delphi, VB oder .Net - Entwickler kann z. B. die  
// „BASS audio library“ verwendet werden  
// Es empfiehlt sich, eine eigene Funktion „play(char* audiofile)“ zu implementieren...  
  
play(„sieben.wav“);  
play(„hundert.wav“);  
play(„drei.wav“);  
play(„und.wav“);  
play(„zwanzig.wav“);
```

Unnützes Wissen ☺

Die Zahl 10^{100} (zehn Sexdezilliarden) wird auch als „Googol“ bezeichnet. Dies entspricht einer 1 mit 100 Nullen. Ein Googol ist damit größer als die Anzahl aller Atome im beobachtenden Universum!

Die Suchmaschine „google“ leitet ihren Namen von dieser unvorstellbar großen Zahl ab, um die Absicht zu verdeutlichen, welche Fülle an Informationen die Suchmaschine im Web finden soll.

Die Zahl 10^{googol} ($= 10^{(10^{100})}$) wird als „Googolplex“ bezeichnet. Hier leitet Google den Namen seines Firmenhauptsitzes „Googleplex“ ab...

Quellen und weitere Informationen

Artikel	Quelle
Zahlennamen	http://de.wikipedia.org/wiki/Zahlennamen/
Vortrag „Zahlen auf Deutsch aussprechen“ von Lothar Schmitz	http://www.informatikjahr.de/algorithmus/
Googol	http://de.wikipedia.org/wiki/Googol