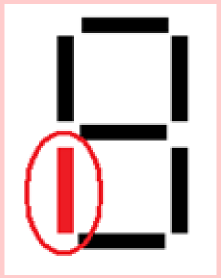
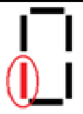
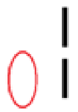
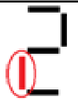
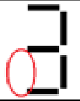
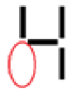
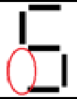
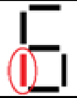
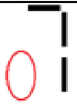
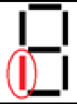
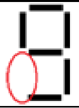
Lecture 5: Binary world of a computer – a closer look and numerical effects

- Review: binary numeral system
- Floating point representation
- Numerical behaviour
- Runtime behaviour (a short look)
- Character codes

Review: binary numeral system

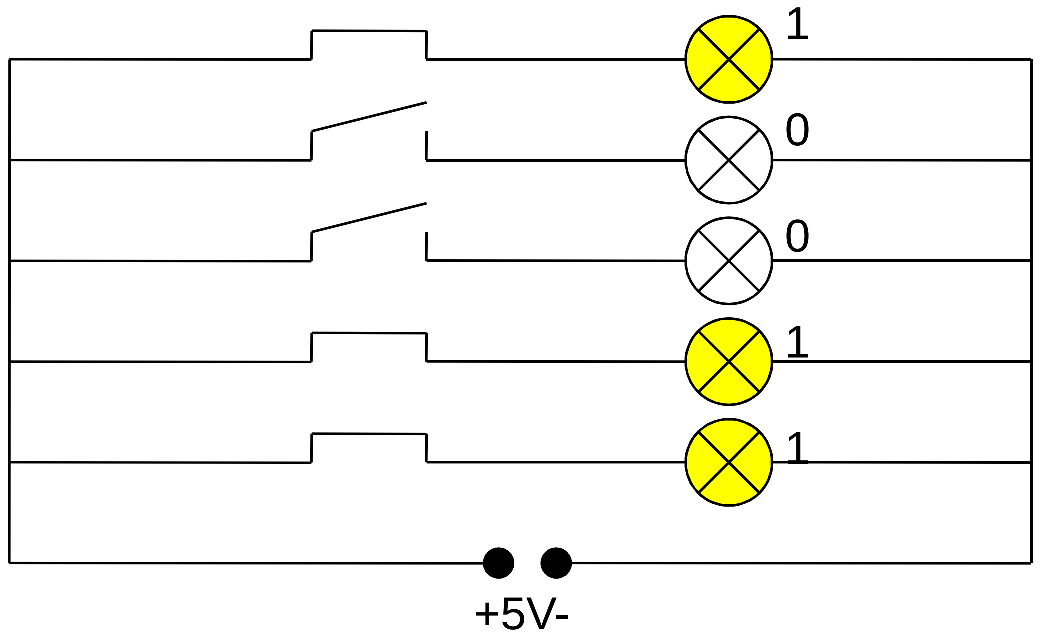


The simplification is also used when a conversion between technical representations of numbers must be done. Now we want to represent the binary numbers from 0_{dez} to 9_{dez} with a 7-segment-display. But we just have a look for one segment segment (see graphic). Give a simplified form for the conversion into that segment (advice for final test: task type could be similar with changed truth table values).

A	B	C	D	DEZ	7-segment-display
0	0	0	0	0	
0	0	0	1	1	
0	0	1	0	2	
0	0	1	1	3	
0	1	0	0	4	
0	1	0	1	5	
0	1	1	0	6	
0	1	1	1	7	
1	0	0	0	8	
1	0	0	1	9	
...					

Review: binary numeral system

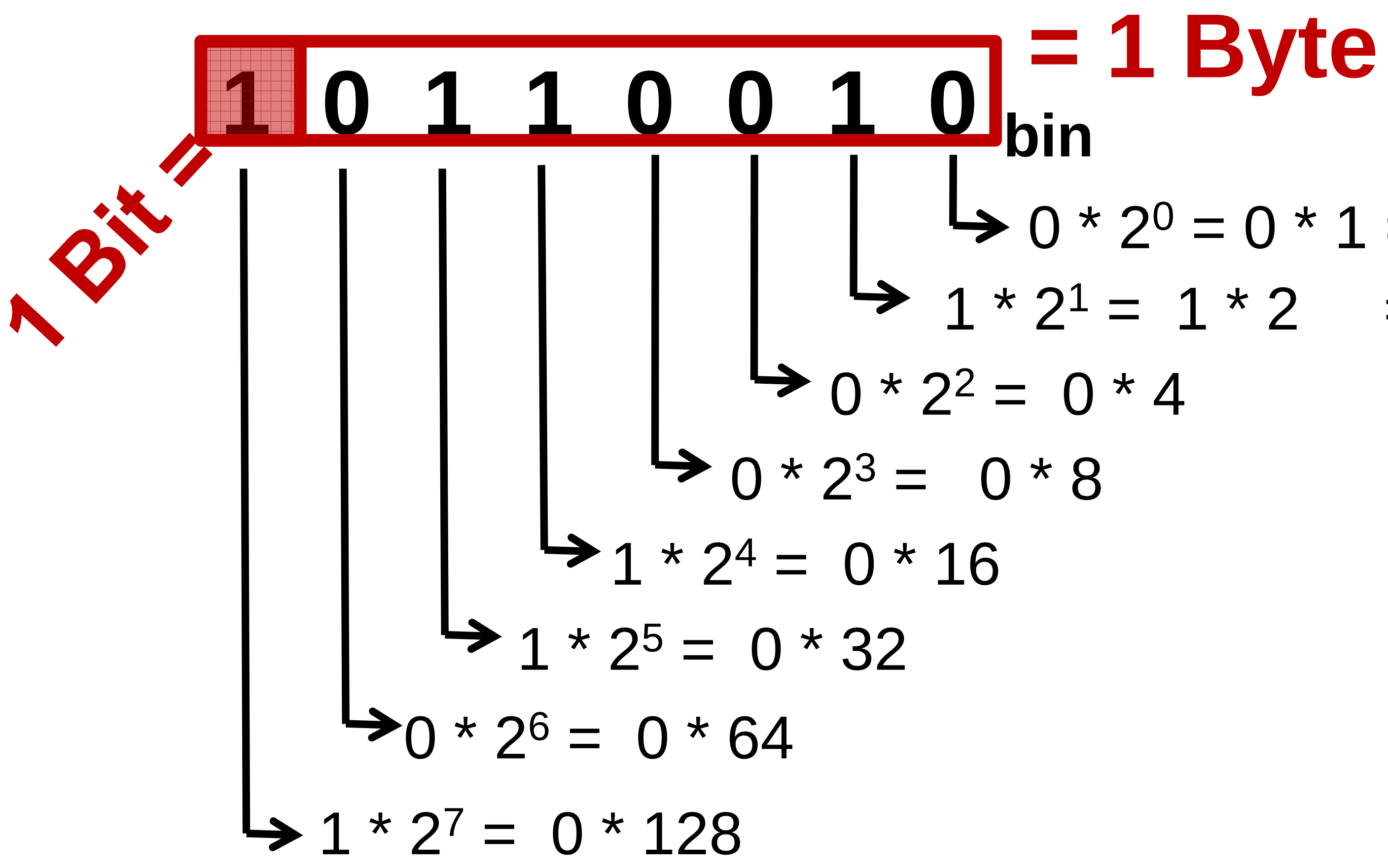
- Origins at China and also developed by the mathematician Leibniz (17th century AD)
- Positional numeral system which represents each number just with 2 symbols, 0 and 1
- These values can be represented by voltage levels in electronic circuits
- For human use very inefficient but with electronic circuits it is possible to create very efficient arithmetic and logic units (ALUs) for the basic operations addition, subtraction, multiplication and division



See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991

Review: binary numeral system

e.g.



$1\ 0\ 1\ 1\ 0\ 0\ 1\ 0_{bin}$

$0 * 2^0 = 0 * 1 = \quad 0_{dec}$

$1 * 2^1 = 1 * 2 \quad = \quad 2_{dec}$

$0 * 2^2 = 0 * 4 \quad = \quad 0_{dec}$

$0 * 2^3 = 0 * 8 \quad = \quad 0_{dec}$

$1 * 2^4 = 0 * 16 \quad = \quad 16_{dec}$

$1 * 2^5 = 0 * 32 \quad = \quad 32_{dec}$

$0 * 2^6 = 0 * 64 \quad = \quad 0_{dec}$

$1 * 2^7 = 0 * 128 \quad = 128_{dec}$

$\sum = 178_{dec}$

Review: binary numeral system

e.g. $\mathbf{1\ 7\ 8}_{\mathbf{dec}}$

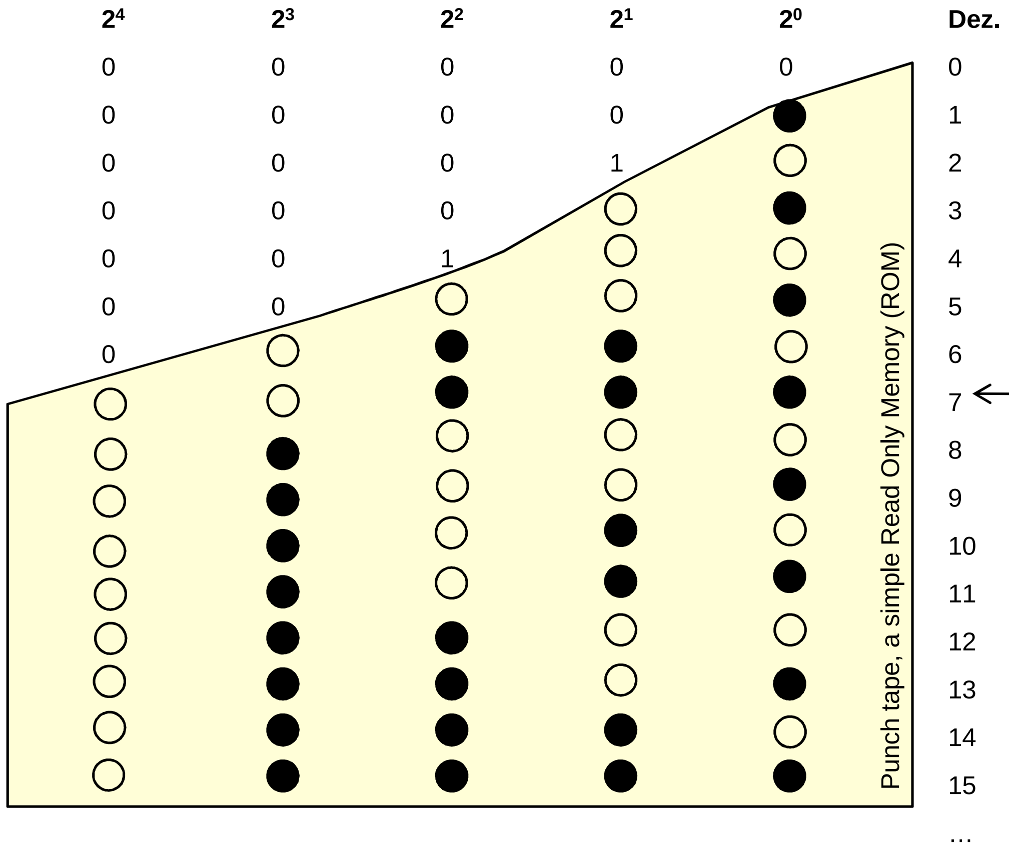
$$\begin{aligned}
\hookrightarrow 178 / 2 &= 89_{dec} \quad \text{Rest } 0_{bin} \\
89 / 2 &= 44_{dec} \quad \text{Rest } 1_{bin} \\
44 / 2 &= 22_{dec} \quad \text{Rest } 0_{bin} \\
22 / 2 &= 11_{dec} \quad \text{Rest } 0_{bin} \\
11 / 2 &= 5_{dec} \quad \text{Rest } 1_{bin} \\
5 / 2 &= 2_{dec} \quad \text{Rest } 1_{bin} \\
2 / 2 &= 1_{dec} \quad \text{Rest } 0_{bin} \\
1 / 2 &= 0_{dec} \quad \text{Rest } 1_{bin}
\end{aligned} \quad \Bigg\uparrow \text{Read direction}$$

$\downarrow$

$\mathbf{1\ 0\ 1\ 1\ 0\ 0\ 1\ 0}_{\mathbf{bin}}$

Review: binary numeral system

Punchcards and the binary numeral system



Punchcards
http://lochkarte.know-library.net/
10.06.2007

Representation of an integer number:

e.g.

$0*2^4 + 0*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 7_{dez}$

Important number systems:

Dual => 2^x
Octal => 8^x
Decimal => 10^x
Hexadecimal => 16^x

See: Rembold, Ulrich et. al.: Einführung in die Informatik für Naturwissenschaftler und Ingenieure. Hanser München Wien 1991

Review: binary numeral system

Basic arithmetic operations on integer numbers

Addition, Subtraction, Multiplication, Division

Addition:

e.g.

$$\begin{array}{r} 17_{10} \\ +\ \ 7_{10} \\ {\scriptstyle 1}\quad \\ 24_{10} \end{array}$$

e.g.

$$\begin{array}{r} 10001_{2} \\ +\ 00111_{2} \\ {\scriptstyle 1\,1\,1}\quad \\ 11000_{2} \end{array}$$

Lecture 5: Binary world of a computer – a closer look and numerical effects

✓ Review: binary numeral system
- Floating point representation
- Numerical behaviour
- Runtime behaviour (a short look)
- Character codes

In cooperation with Karin Hedman
karin.hedman@bv.tum.de

Floating point representation

Standard IEEE 754:

e.g. $1 = +\,0.1 \times 10^{+1}$

$3.1415926 = +\,0.31415926 \times 10^{+1}$

$$\begin{tikzcd}
\text{Sign(s) x} \arrow[d] & \text{Mantissa(m)} \arrow[d] & \text{x Base(b)} \arrow[d] & {}^{\text{+/-Exponent(e)}} \arrow[d] \\
(-1)^{v} & \times\,(\overbrace{m_{t-1},m_{t-2},\dots,m_1,m_0})_2 & \times\,2 & {}^{(\overbrace{e_{s-1},e_{s-2},\dots,e_1,e_0})_2-\text{Offset}}
\end{tikzcd}$$

e.g. $1 = (-1)^0 \times (1)_2 \times 2^{01111111}$

<u>This means as consequence for floating point numbers:</u>
- **Approximation of a real number**
- **Set of floating point numbers is a finite subset of the rational numbers**
- **Together with the on them defined operations they build a finite arithmetic**

Floating point representation

Basic arithmetic operations on floating point numbers

Addition, Subtraction, Multiplication, Division

Addition: e.g.

$$3.46620 \times 10^{12}_{10}$$
$$+\ 0.211900 \times 10^{-2}_{10}$$

$$0.346620 \times 10^{13}_{10}$$
$$+\ 0.211900 \times 10^{-2}_{10}$$

$$0.346620 \times 10^{13}_{10}$$
$$+\ 0.00000000000000211900 \times 10^{13}_{10}$$

$$0.346620 \times 10^{13}_{10}$$
$$+\ 0.000000\boxed{00000000211900} \times 10^{13}_{10}$$

Integer-Addition
(see slide before)

Problem:
The second number is so small that it only will be representated as 0 with given decimal places. So the addition doesn‘t change the result even when the second number is added very often!
=> absorption-problem

[1] See: Precht, Manfred; Meier, Nikolaus; Kleinhein, Joachim: EDV-Grundwissen. Eine Einführung in Theorie und Praxis der modernen EDV. Addison-Wesley (Deutschland) GmbH Bonn 1994

Lecture 5: Binary world of a computer – a closer look and numerical effects

- ✓ Review: binary numeral system
- ✓ Floating point representation
- Numerical behaviour
- Runtime behaviour (a short look)
- Character codes

In cooperation with Karin Hedman
karin.hedman@bv.tum.de

Floating point representation

Attributes of floating point numbers:

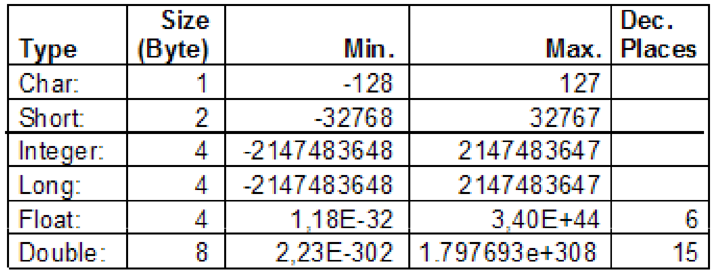
- Cancellation: Subtraktion of numbers with equal dimensions produces wrong results
- Absorption: The addition or subtraction of a big number and a very small number doesn‘t change the big number
- Underflow: Numbers lower than minimal presentable floating point numbers becomes 0
- Invalid associative law: (x + y) + z = x + (y + z)
- Invalid distributive law: x (y + z) = (xy) + (xz)
- Solutions for not solvable equation systems can be found
- Konversion inaccuracies between decimal and dual system
- Representation problems of simple decimal numbers (0.1 is sometimes represented as 0.09999..)
- Risk of deterministic chaos while using iterative calculations

[1] See: http://de.wikipedia.org/wiki/Gleitkommazahl, 10.06.2007

Numerical behaviour – the limitative world for representation

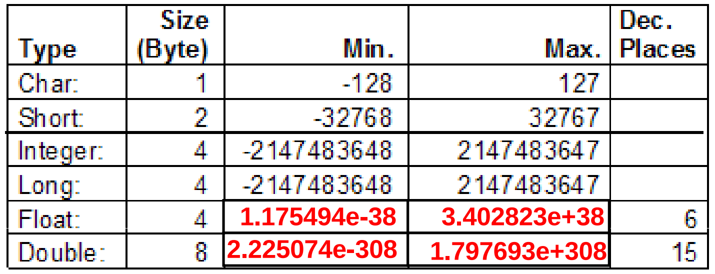
Types and their limitations

Windows XP (MSVC++)



Type	Size (Byte)	Min.	Max.	Dec. Places
Char:	1	-128	127	
Short:	2	-32768	32767	
Integer:	4	-2147483648	2147483647	
Long:	4	-2147483648	2147483647	
Float:	4	1,18E-32	3,40E+44	6
Double:	8	2,23E-302	1.797693e+308	15

Debian Linux (G++)



Type	Size (Byte)	Min.	Max.	Dec. Places
Char:	1	-128	127	
Short:	2	-32768	32767	
Integer:	4	-2147483648	2147483647	
Long:	4	-2147483648	2147483647	
Float:	4	**1.175494e-38**	**3.402823e+38**	6
Double:	8	**2.225074e-308**	**1.797693e+308**	15

Not all numbers can be represented, e.g. 2/3 = 0,666667 (Float)
(Dependent on computer operating system)

This means as consequence for floating point numbers: Roundoff errors
=> But are these errors a problem, when we are only interested in a view decimal places after the decimal point?

Numerical behaviour – an experiment

The Lorentz – experiment (1956)[1]

Simulation of meteorological forecast methodes with 12 equations which have none-periodic solutions
Verification of some intermediate data with a small computer
Results of a previous calculation as starting conditions for a following
The solutions changed, the computer has changed it‘s behaviour
The rounding of 6 decimal places to 3 caused an effect with a dimension of signal strength when 2 month models were calculated
Observation/Calculation inaccuracies grow very fast (exponentially)

=> Longterm forecasts aren‘t possible even when the models would be perfect

[1] See: Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar: Bausteine des Chaos. Fraktale. Rowohlt Taschenbuch Verlag GmbH Hamburg 1998 (Orig.: Fractals for the Classroom. Part 1. Springer Verlag New York 1992)

Numerical behaviour – an experiment

A simple equivalent to the Lorentz – experiment[1]: Verhulsts logistic model of population dynamics

$$\boxed{p_{n+1} = p_n + rp_n\,(1-p_n)}$$

Prediction of a population in a limited habitat (e.g. some organisms in a petri dish)
The expansion rate depends on the current population based on the maximal population
The expansion rate at time n is proportional to the difference of current and maximal population (degree of habitat which is not yet populated)

[1] See: Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar: Bausteine des Chaos. Fraktale. Rowohlt Taschenbuch Verlag GmbH Hamburg 1998 (Orig.: Fractals for the Classroom. Part 1. Springer Verlag New York 1992)

Numerical behaviour – an experiment

The problem with Verhulsts logistic model of population dynamics in the limitative world of the computer[1]

E.g.: starting value for p is 0,01 (r = 3)

p_n	$p_{n+1} = p_n + rp_n (1-p_n)$
0,010000000000000	0,039700000000000
0,039700000000000	0,154071730000000
0,154071730000000	0,545072626044421
0,545072626044421	

The amount of necessary decimal places, to represent the correct result, grows very fast (exponential).
=> But in a computer, there are limited type representations
=> Roundoff erros
=> Error propagation

[1] See: Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar: Bausteine des Chaos. Fraktale. Rowohlt Taschenbuch Verlag GmbH Hamburg 1998 (Orig.: Fractals for the Classroom. Part 1. Springer Verlag New York 1992)

Numerical behaviour – an experiment

Let’s compare FLOAT (6 dec. places) - and DOUBLE (15 dec. places)-representations

E.g.: starting value for p is 0,01 (r = 3)

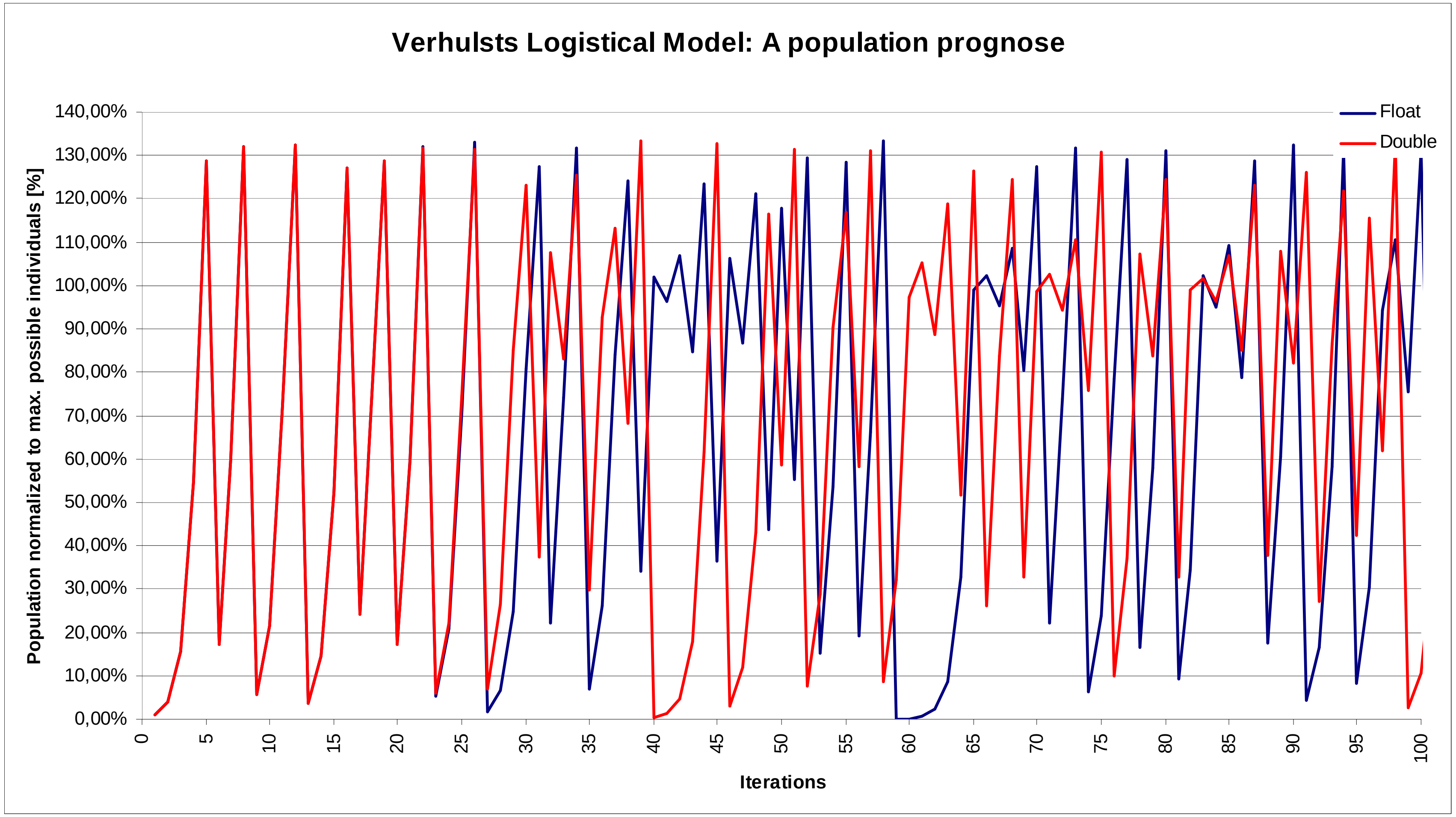
The program:

```
int main ()
{
    float fP = 0.01;
    float fR = 3.0;
    double dP = 0.01;
    double dR = 3.0;
    unsigned long ulIteration = 0;

    while (1)
    {
        vPrintResult (ulIteration, fP, dP);
        fP = fP + fR * fP * (1 - fP);
        dP = dP + dR * dP * (1 - dP);
        ulIteration++;
        if (ulIteration == 10001)
            break;
    }

    return 0;
}
```

ITERATION	FLOAT	DOUBLE	DIFFERENCE
0	1,00%	1,00%	0,00%
1	3,97%	3,97%	0,00%
2	15,41%	15,41%	0,00%
3	54,51%	54,51%	0,00%
4	128,90%	128,90%	0,00%
5	17,15%	17,15%	0,00%
6	59,78%	59,78%	0,00%
7	131,91%	131,91%	0,00%
8	5,63%	5,63%	0,00%
9	21,56%	21,56%	0,00%
10	72,29%	72,29%	0,00%
11	132,38%	132,38%	0,00%
12	3,77%	3,77%	0,00%
13	14,65%	14,65%	0,00%
14	52,16%	52,17%	0,00%
15	127,02%	127,03%	0,00%
16	24,05%	24,04%	-0,01%
17	78,84%	78,81%	-0,03%
18	128,89%	128,91%	0,02%
19	17,19%	17,11%	-0,08%
20	59,90%	59,65%	-0,25%
21	131,96%	131,86%	-0,10%
22	5,44%	5,84%	0,40%
23	20,86%	22,33%	1,47%
24	70,38%	74,36%	3,98%
25	132,92%	131,56%	-1,36%
26	1,65%	7,00%	5,36%
27	6,50%	26,54%	20,04%
28	24,74%	85,04%	60,29%
29	80,61%	123,21%	42,60%
30	127,50%	37,41%	-90,09%

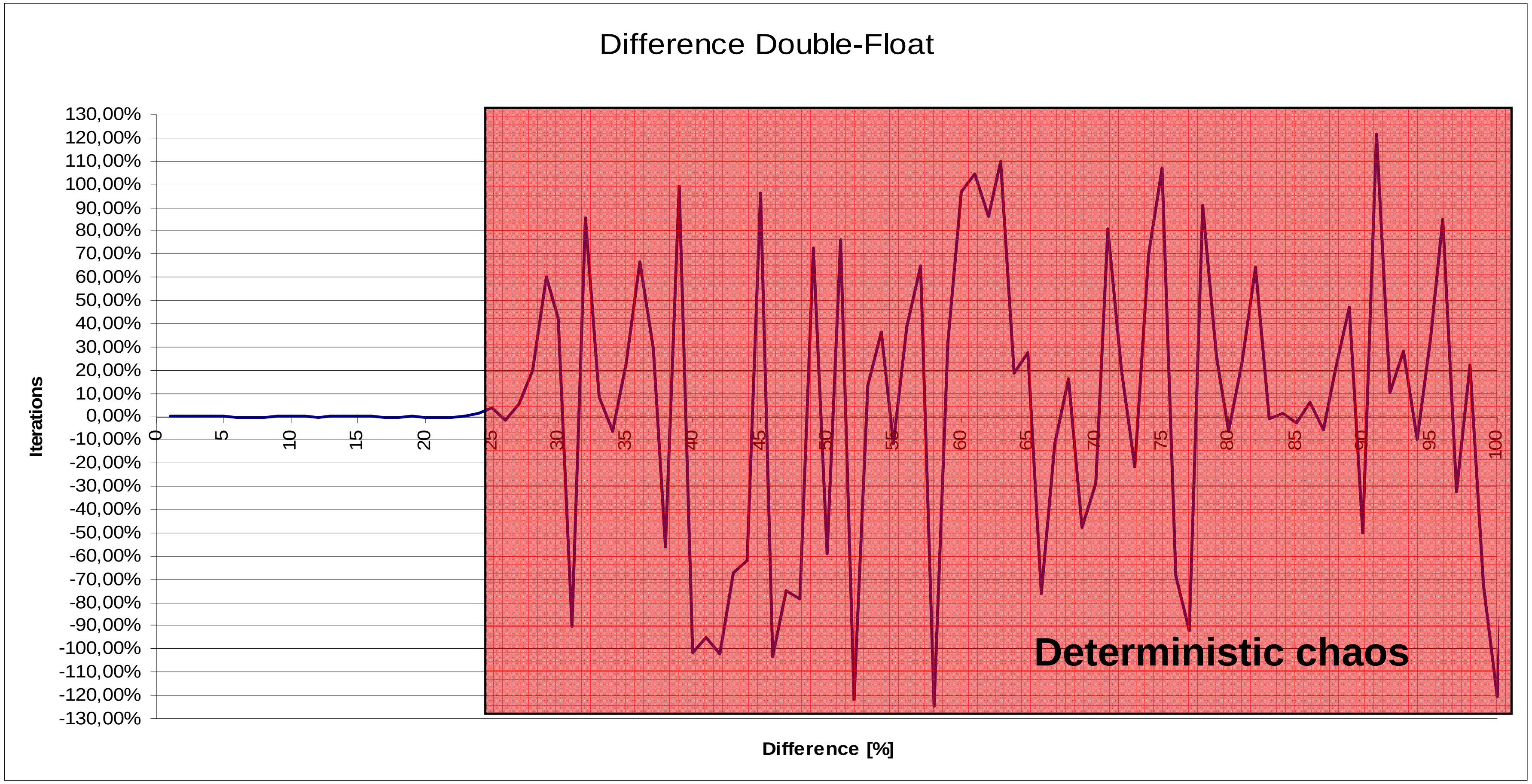
Numerical behaviour – an experiment

Let’s compare FLOAT (6 dec. places) – and DOUBLE (15 dec. places)-representations



Numerical behaviour – an experiment

Let’s compare FLOAT (6 dec. places) – and DOUBLE (15 dec. places)-representations



Numerical behaviour – an experiment

Even the starting conditions can‘t be represented equaly:

	FLOAT-Representation	DOUBLE-Representation
0,01	0,00999999977648258	0,01

First conclusion: Do not mix types!

But there are possibilities of mistakes (original example):

```
...
      SUBROUTINE SUNTRUE(TC,SUNLONG,OBL)
C       ==================================
C
C************************************************************************
C         CALCULATES TRUE LONGITUDE OF SUN RELATIVE TO MEAN EQUINOX OF
C         DATE, AND OBLIQUITY OF ECLIPTIC (OBL).
C         INPUT  : TC - JULIAN CENTURIES SINCE 1900.0
C         OUTPUT : SUNLONG - TRUE LONG. OF SUN IN RADIANS
C                OBL     - OBLIQUITY IN RADIANS
C************************************************************************
C
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /MATH/ PIO2, PI, TWOPI, RDPDG, RDPHR, RDPAS, RDPTS
C
C-----------------------------------------------------------------------
C         SUN'S MEAN ANOMALY(ANOM), LONGITUDE OF PERIGEE(PER) & OBLIQUITY
C         FROM FORMULAE IN EXPLANATORY SUPPLEMENT
C-----------------------------------------------------------------------
C
      ANOM=(358.475833D0+35999.04975D0*TC- 0.00015D0*TC*TC)*RDPDG
      PER =(281.220844D0+   1.719175D0*TC-0.000453D0*TC*TC)*RDPDG
      OBL =( 23.452294D0-  0.0130125D0*TC-.0000016D0*TC*TC)*RDPDG
      ECCEN=0.01675104D0-0.00004180*TC
C
C-----------------------------------------------------------------------
C         TRUE ANOMALY FROM MEAN ANOMALY USING EQUATION OF THE CENTRE
C         AND THEN ADD TRUE ANOMALY TO LONGITUDE OF PERIGEE
C-----------------------------------------------------------------------
C
      TRUE=ANOM+(2.0*ECCEN-0.25*(ECCEN**3))*DSIN(ANOM)+1.25*(ECCEN**2)*
     @     DSIN(2.0D0*ANOM)+1.083333*(ECCEN**3)*DSIN(3.0D0*ANOM)
      SUNLONG=TRUE+PER
      RETURN
      END
...
```

Fortran-compiler dependent handling of conversion between float and double
(D = double, all of the others is float)

(Original NASA-code to calculate SLR satellite passages)

Numerical behaviour – an experiment

Another problem shown with Verhulsts logistic model of population dynamics in the limited world of the computer[1]

The following mathematically equivalent representations of the Verhulst equation doesn‘t create the equivalent results in the computer even if you use the same type representation:

Mathematical

$$p_{n+1} = p_n + r\, p_n\, (1-p_n)$$

$$=$$

$$p_{n+1} = (1 + r)\, p_n - r\, p_n^2$$

$$=$$

$$p_{n+1} = (1 + r)\, p_n - r\, p_n\, p_n$$

[1] See: Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar: Bausteine des Chaos. Fraktale. Rowohlt Taschenbuch Verlag GmbH Hamburg 1998 (Orig.: Fractals for the Classroom. Part 1. Springer Verlag New York 1992)

Numerical behaviour – an experiment

Another problem shown with Verhulsts logistic model of population dynamics in the limited world of the computer[1]

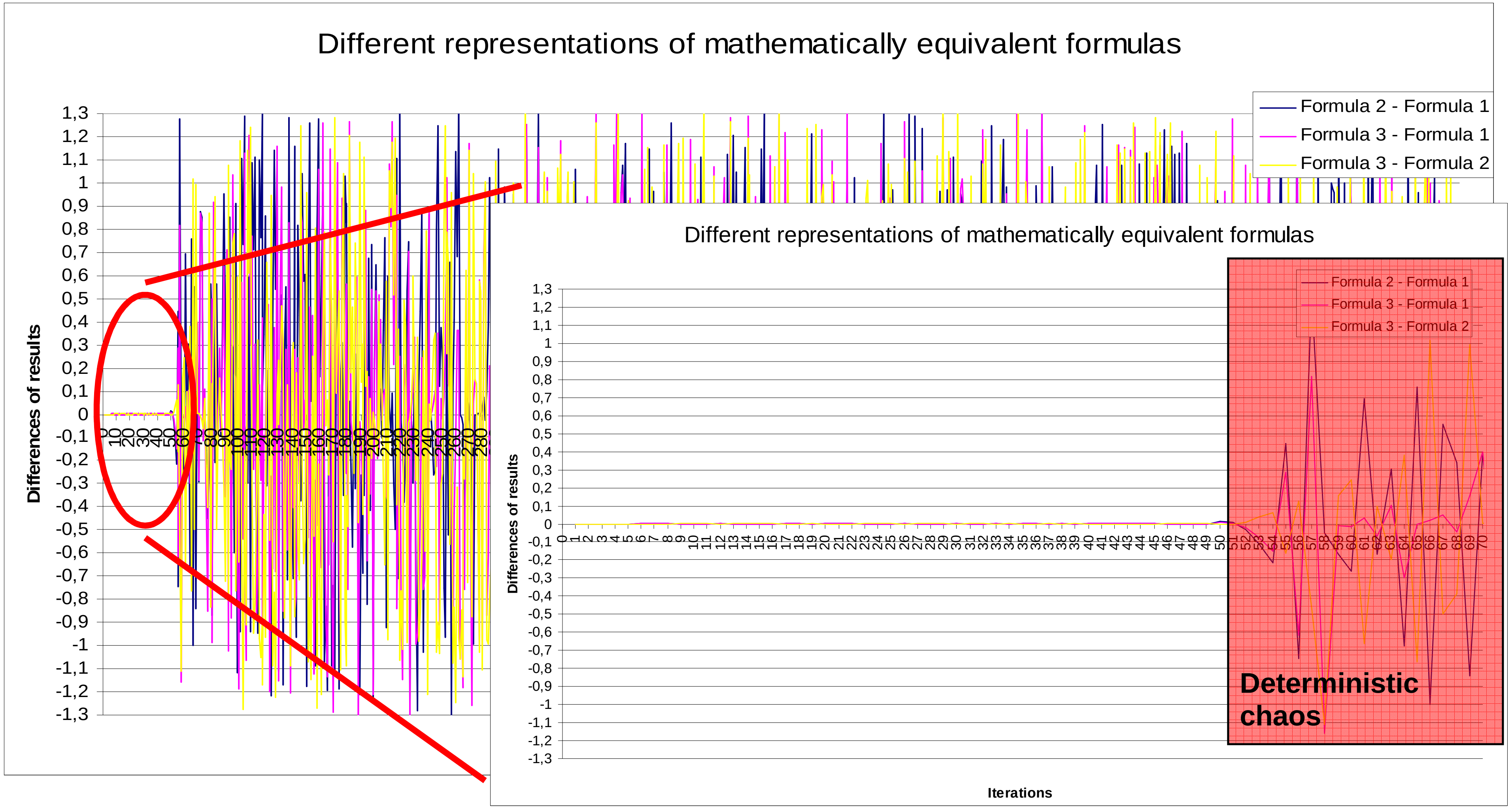
E.g.: starting value for p is 0,01 (r = 3)

The program:

```
int main ()
{
    double dP1 = 0.01;
    double dP2 = 0.01;
    double dP3 = 0.01;
    double dR = 3.0;
    unsigned long ulIteration = 0;

    while (1)
    {
        vPrintResult (ulIteration, dP1, dP2, dP3);
        dP1 = dP1 + dR * dP1 * (1 - dP1);
        dP2 = (1 + dR) * dP2 - dR * pow(dP2,2);
        dP3 = (1 + dR) * dP3 - dR * dP3 * dP3;
        ulIteration++;
        if (ulIteration == 10001)
            break;
    }

    return 0;
```

Numerical behaviour – an experiment

Another problem shown with Verhulsts logistic model of population dynamics in the limited world of the computer[1]



Numerical behaviour – an experiment

Another problem shown with Verhulsts logistic model of population dynamics in the limited world of the computer[1]

The following mathematically equivalent representations of the Verhulst equation doesn‘t create the equivalent results in the computer even if you use the same type representation:

$$\begin{tikzcd}
\underline{\text{Mathematical}} & & \underline{\text{Numerical}} & \\
& p_{n+1} = p_n + r\, p_n\, (1-p_n) \arrow[dl] \arrow[dr] & & \\
= \arrow[dr] & & \cancel{=} \arrow[dl] & ? \\
& p_{n+1} = (1 + r)\, p_n - r\, p_n^2 \arrow[dl] \arrow[dr] & & \\
= \arrow[dr] & & \cancel{=} \arrow[dl] & \\
& p_{n+1} = (1 + r)\, p_n - r\, p_n\, p_n & &
\end{tikzcd}$$

Numerical behaviour – an experiment

Third conclusion: Try to avoid iterative feedbacks!

=> But it's not always possible, e.g. Verhulsts model

For example:
It is better to use

$$t_n = t_0 + \ n * \quad t$$

than

$$t_n = t_0$$
$$t_{n+1} = t_n + \quad t$$

Fourth conclusion: Do not simply accept "black box" - algorithms!

=> Iterativ algorithms with floating point numbers are always sensitiv dependent on the starting conditions

For example use well known and described solutions given at:
Press, William H.; Teukolsky, Saul A.; Vetterling, William T.; Flannery, Brian P.: Numerical Recipes in C++. The Art of Scientific Computing. Second Edition. Cambridge University Press New York 2002

Numerical behaviour – an experiment

Conclusion:

And even if you care about all of such things and if you use the real world most adapting model, you can‘t trust each results for predictions for the future! This means, that it is not possible to predict iterativly any longlasting timeperiode into future with modern, but deterministic data representing computers! And it is difficult to do a good error analysis (because of the chaotical effects).

→ **Overall conclusion: Each deterministic number representation generates from one not clearly defined iteration on (depends on start condition, algorithm, etc.) deterministic chaos, which influences significant decimal places.**

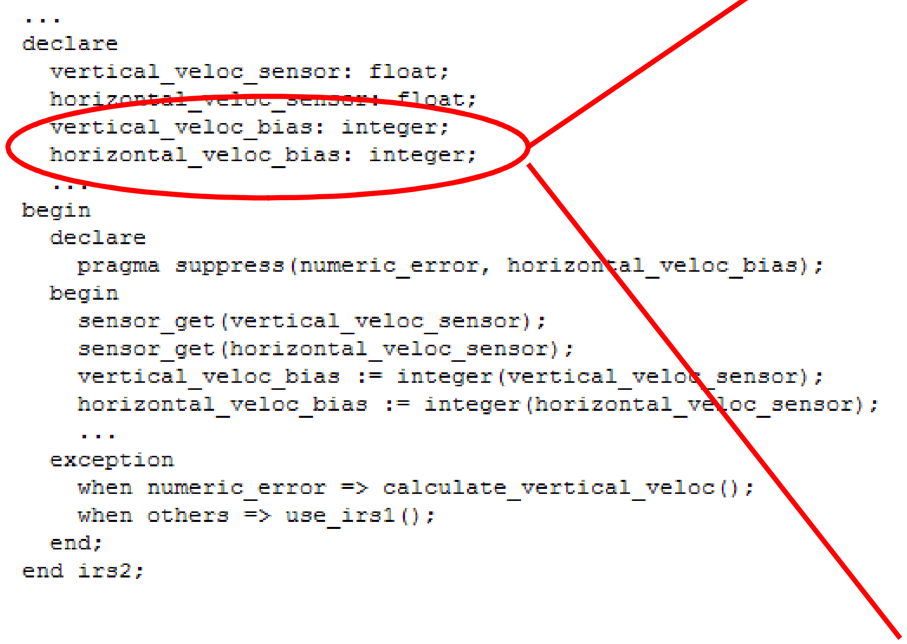
[1] See: Peitgen, Heinz-Otto; Jürgens, Hartmut; Saupe, Dietmar: Bausteine des Chaos. Fraktale. Rowohlt Taschenbuch Verlag GmbH Hamburg 1998 (Orig.: Fractals for the Classroom. Part 1. Springer Verlag New York 1992)

Famous problems ...

Disasterous design errors: Ariane 5 (Kourou, 04.06.1996)

Maiden flight of a new European carrier rocket

Part of the original Navigation-ADA-Code:



```
...
declare
  vertical_veloc_sensor: float;
  horizontal_veloc_sensor: float;
  vertical_veloc_bias: integer;
  horizontal_veloc_bias: integer;
  ...
begin
  declare
    pragma suppress(numeric_error, horizontal_veloc_bias);
  begin
    sensor_get(vertical_veloc_sensor);
    sensor_get(horizontal_veloc_sensor);
    vertical_veloc_bias := integer(vertical_veloc_sensor);
    horizontal_veloc_bias := integer(horizontal_veloc_sensor);
    ...
  exception
    when numeric_error => calculate_vertical_veloc();
    when others => use_irs1();
  end;
end irs2;
```

The problem:
37 seconds after ignition (30 seconds after liftoff) at a height of 3700 meters Ariane 5 reaches a vertical velocity of 32768.0 (internal units), which was 5 times higher than given by Ariane 4. The consequence: the cast into an integer number resulted in an overrun, which was not caught by the software.
The software was also located at the two redundant control systems, so that an irreparable error was emerged, which turned of one of the control systems and brought the other because of wrong diagnosis data into a correction phase to correct the trajectory. The senseless control commands to the busters to correct the 20 degree flight path difference brought the rocket into an instable state so that the self-destruction mechanism was flushed.

Financial and psychological damage:
125 million Euro launch costs, 425 million Euro for 4 cluster satellites, 300 million Euro additional developements
2 – 3 years suspension of orders (the first commercial flight was 1999)

(Reference: http://www-aix.gsi.de/~giese/swr/ariane5.html, Download 14.12.2006)

Floating point

What does the IEEE 754 describe?
Give a schematic representation!

Give four floating point attributes in a numerical limited system.

Give a short definition for deterministic chaos and explain an example where it can easily appear in a deterministic, numerical computer world.

Give four number systems (including the bases) used in computer science?

Lecture 5: Binary world of a computer – a closer look and numerical effects

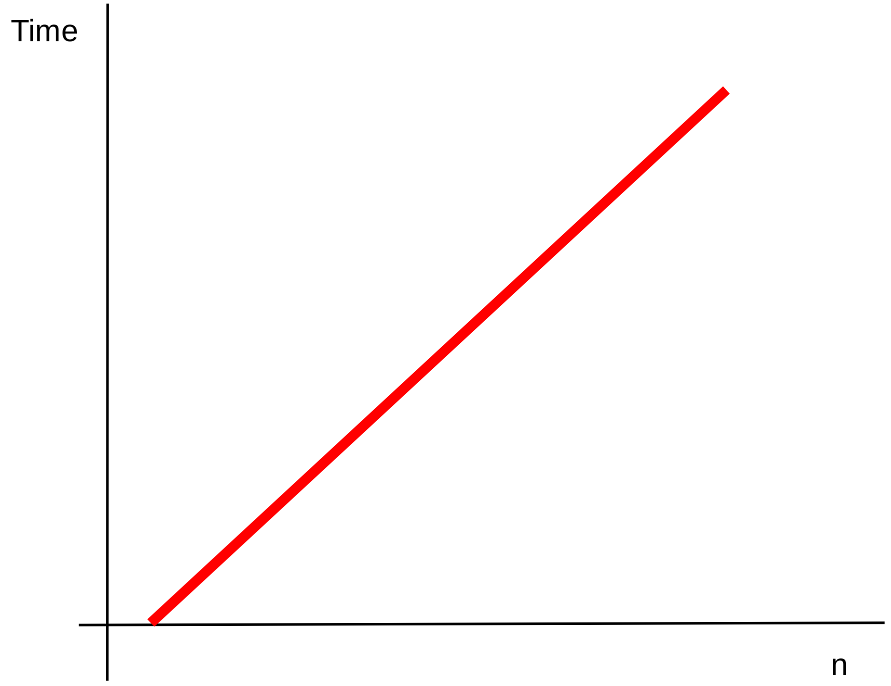
- ✓ Review: binary numeral system
- ✓ Floating point representation
- ✓ Numerical behaviour
- - Runtime behaviour (a short look)
- - Character codes

In cooperation with Karin Hedman
karin.hedman@bv.tum.de

Runtime behaviour

Sum of the first n natural numbers

e.g. n=1: 1 = 1	0 additions
n=2: 1+2 = 3	1 addition
n=3: 1+2+3 = 6	2 additions
n=4: 1+2+3+4 = 10	3 additions
…	…



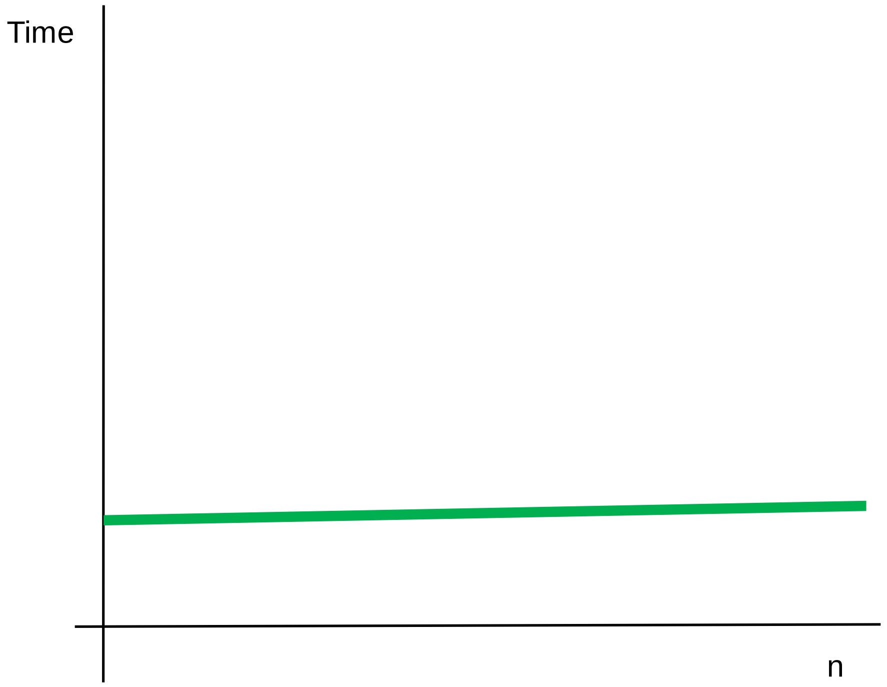
Reference: Singh, Simon: Fermat‘s Last Theorem. Fourth Estate, London 1997

Runtime behaviour

Sum of the first n natural numbers with one formula

sum(n) = 0.5 n (n + 1)

1 addition
2 multiplications



Reference: Singh, Simon: Fermat‘s Last Theorem. Fourth Estate, London 1997

Runtime behaviour

Sum of the first n natural numbers with one formula

sum(n) = 0.5 * n * (n + 1)

1 addition
2 multiplications

Proof : Mathematical induction

1. Step: n = 1: sum(1) = 0.5 * 1 * (1 + 1) = 1

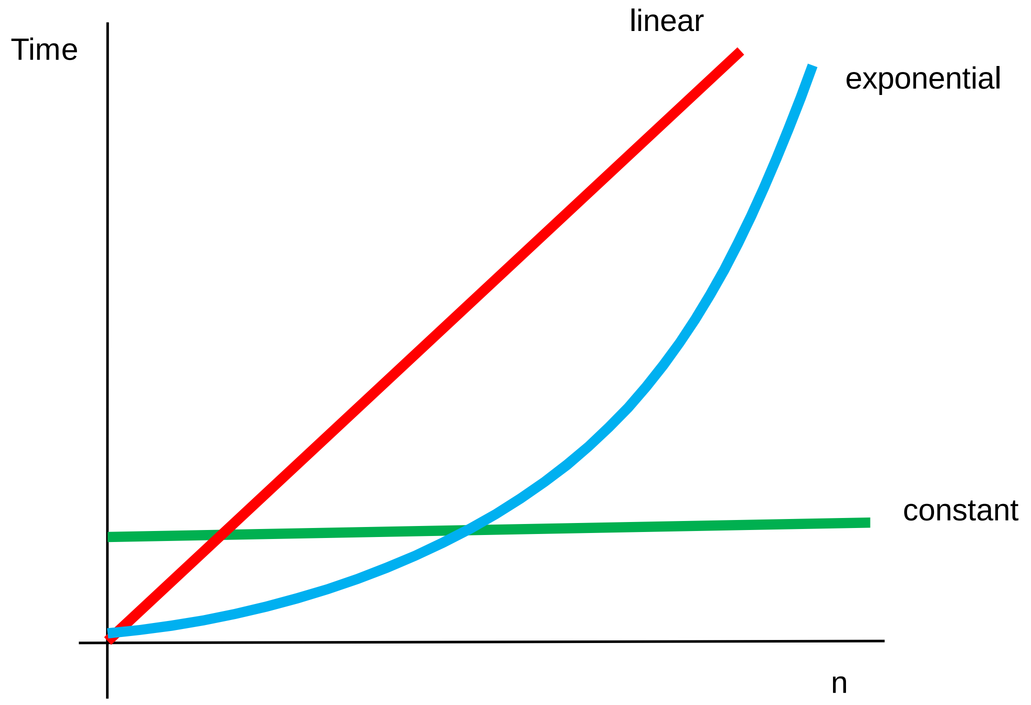
2. Step: n = n+1

sum(n+1) = sum(n) + (n + 1)
sum(n+1) = 0.5 * n * (n + 1) + (n + 1)
sum(n+1) = 0.5 * (n + 1) * ((n + 1) + 1)
=> n = n+1

Reference: Singh, Simon: Fermat‘s Last Theorem. Fourth Estate, London 1997

Runtime behaviour

Problemcategories:



Runtime behaviour

Matlab: Measure code efficiency

In older Matlab versions „flops“ counted number of floating point operations (obsolete function since version 6)

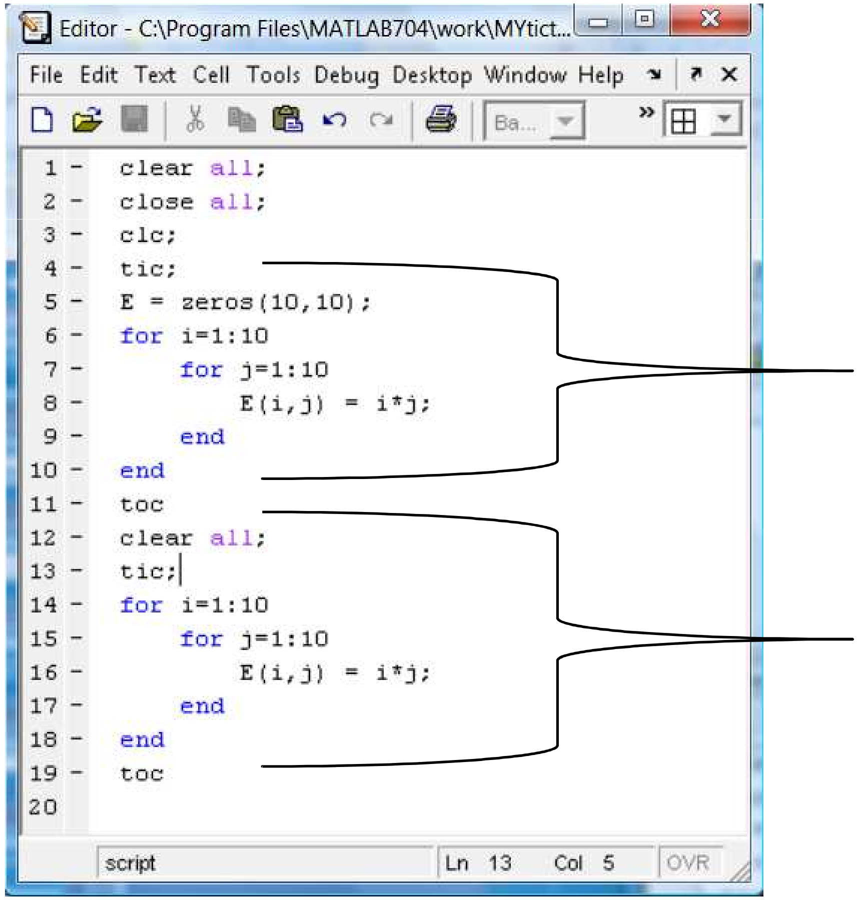
Measure the run time with „tic“ and „toc“
e.g.

```
A = [0.8 0.3; 0.2 0.7];
E = zeros(2,10);
tic;
for k=1:10
   E(:,k) = eig(A^k);
end
toc;
   Elapsed time is 0.005646 seconds
```

Runtime behaviour

Matlab: Initialisation of matrices

Initialise the matrices before usage can improve timing e.g.


Editor - C:\Program Files\MATLAB704\work\MYtict...
File Edit Text Cell Tools Debug Desktop Window Help
script
Ln 13 Col 5
OVR

```
clear all;
close all;
clc;
tic;
E = zeros(10,10);
for i=1:10
    for j=1:10
        E(i,j) = i*j;
    end
end
toc
clear all;
tic;
for i=1:10
    for j=1:10
        E(i,j) = i*j;
    end
end
toc
```

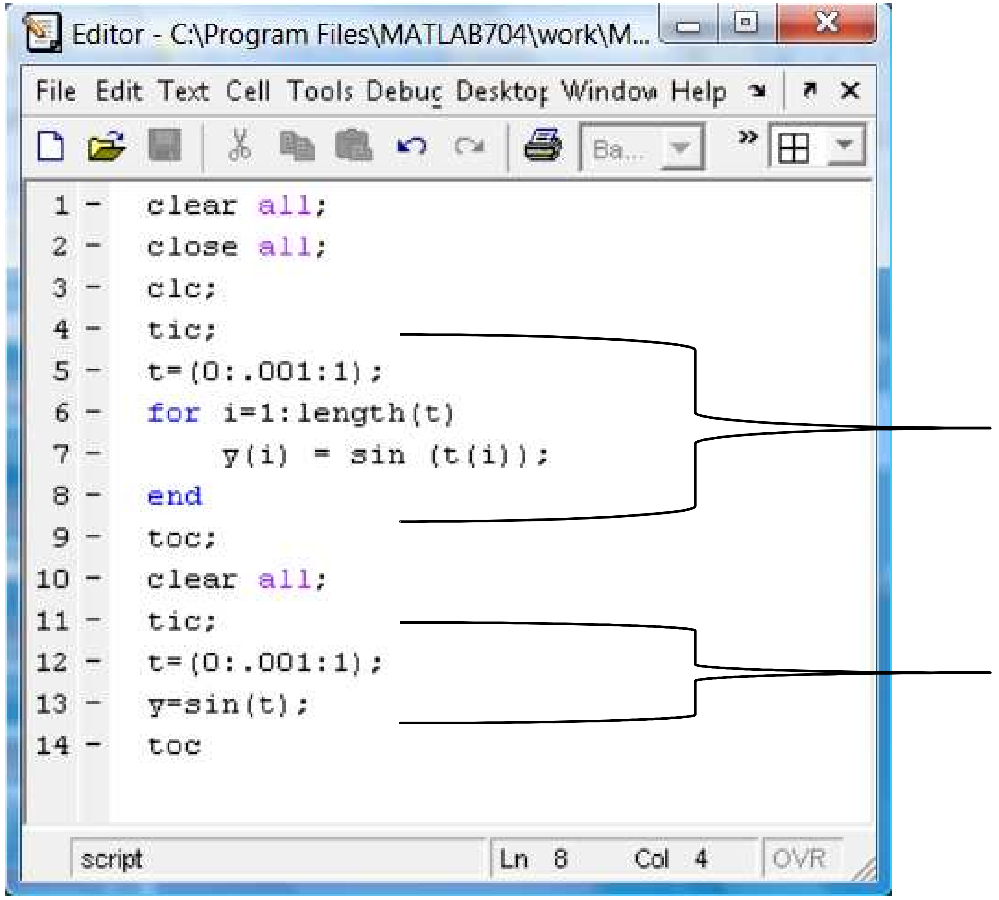
Elapsed time is 0.000026 seconds.

Elapsed time is 0.000063 seconds.
(Loop lasts longer because Matlab has to increase matrix iteratively combined with memory allocation)

Runtime behaviour

Matlab: Vectorisation of operations

Matlabs functions are optimized for matrices so use matrices/ vectors as input e.g.


Editor - C:\Program Files\MATLAB704\work\M...
File Edit Text Cell Tools Debug Desktop Window Help
Ba...
script
Ln 8 Col 4
OVR

```
clear all;
close all;
clc;
tic;
t=(0:.001:1);
for i=1:length(t)
    y(i) = sin (t(i));
end
toc;
clear all;
tic;
t=(0:.001:1);
y=sin(t);
toc
```

Elapsed time is 0.031466 seconds.

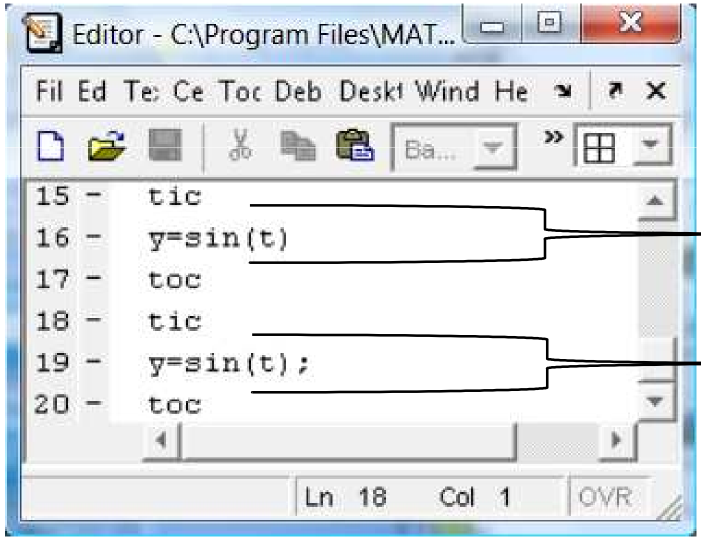
Elapsed time is 0.000219 seconds.

Runtime behaviour

Matlab: Try to avoid …

- Loops => better use Matlab functions or optimized algorithms

- Input / output to command window or to file


Editor - C:\Program Files\MAT...
Fil Ed Te: Ce Toc Deb Deskt Wind He
Ba...
Ln 18 Col 1
OVR

```
tic
y=sin(t)
toc
tic
y=sin(t);
toc
```

Elapsed time is 0.005052 seconds.

Elapsed time is 0.000157 seconds.

(But this rules are not always valid: basic operations could be faster with a classical programming style)

Runtime behaviour

Matlab: Usage of spares matrices

In performing matrix computations, MATLAB normally assumes that a matrix is dense; that is, any entry in a matrix may be nonzero. If, however, a matrix contains sufficiently many zero entries, computation time could be reduced by avoiding arithmetic operations on zero entries and less memory could be required by storing only the nonzero entries of the matrix. This increase in efficiency in time and storage can make feasible the solution of significantly larger problems than would otherwise be possible. MATLAB provides the capability to take advantage of the sparsity of matrices.

Matlab has two storage modes, full and sparse, with full the default. The functions full and sparse convert between the two modes. For a matrix A, full or sparse, nnz(A) returns the number of nonzero elements in A.

```
e.g.:
   A=[0 0 1; 1 0 2;0 -3 0];
   S=sparse(A)
```

Lecture 5: Binary world of a computer – a closer look and numerical effects

- ✓ Review: binary numeral system
- ✓ Floating point representation
- ✓ Numerical behaviour
- ✓ Runtime behaviour (a short look)
- - Character codes

In cooperation with Karin Hedman
karin.hedman@bv.tum.de

Code

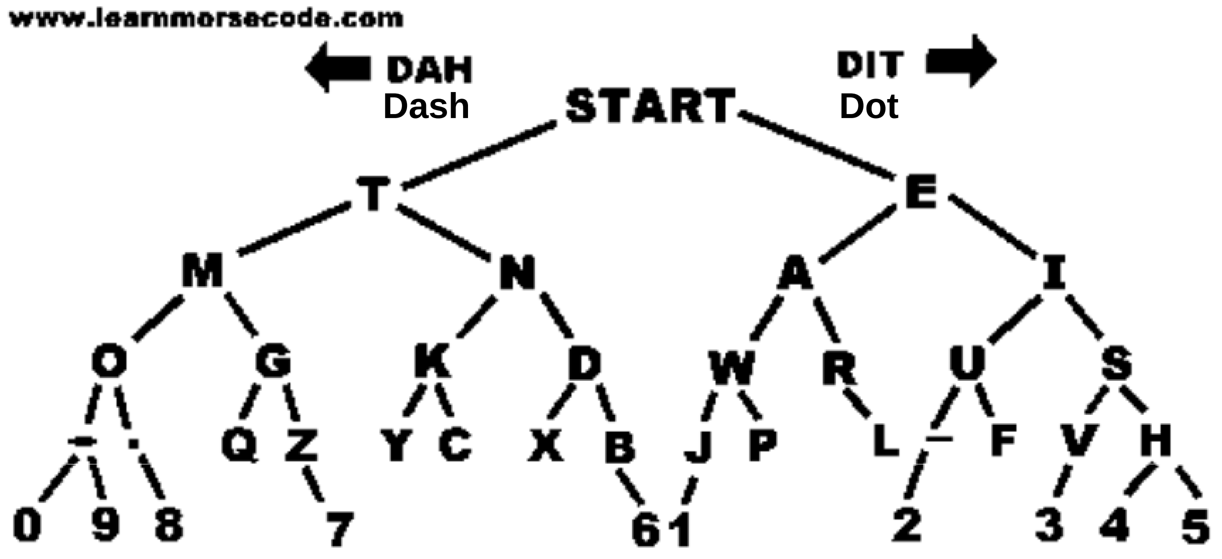
Definition:

Langenscheidt (freely translated):
Aggrement on a character system for communication, data processing and data transfer[1]

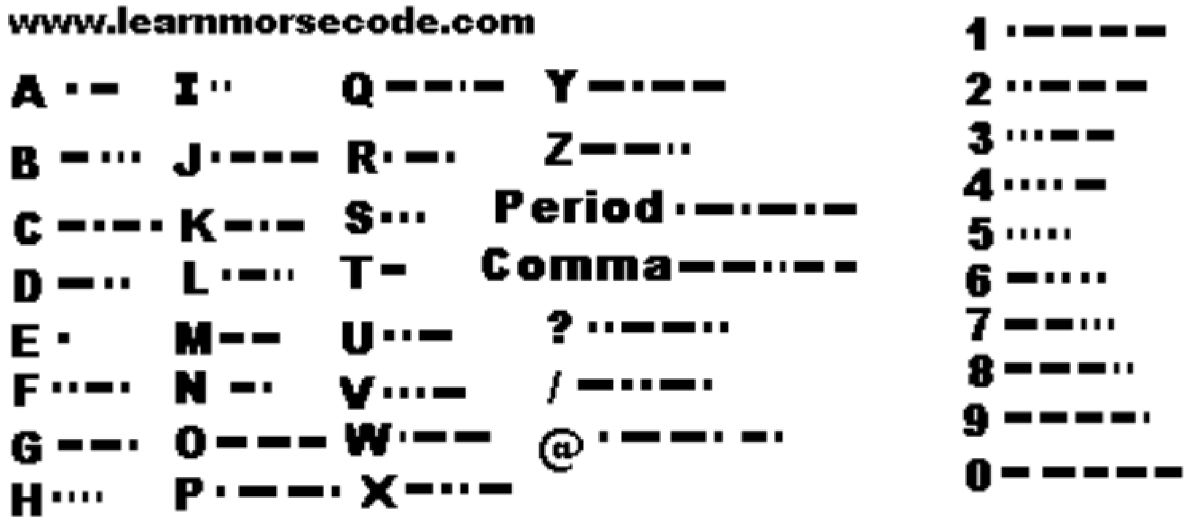
A code is an unique specification to transform a set of signs into another set of signs [2]

[1] See: http://services.langenscheidt.de/cgi-bin/fremdwb/searchfw.pl, Download 30.11.2008
[2] See: Precht, Manfred; et. Al.: EDV-Grundwissen. Addison-Wesley 1994

Character codes history: Morse code



„Binary“ code (plus pauses of three types) with variable code word length for telegraphy


www.learnmorsecode.com

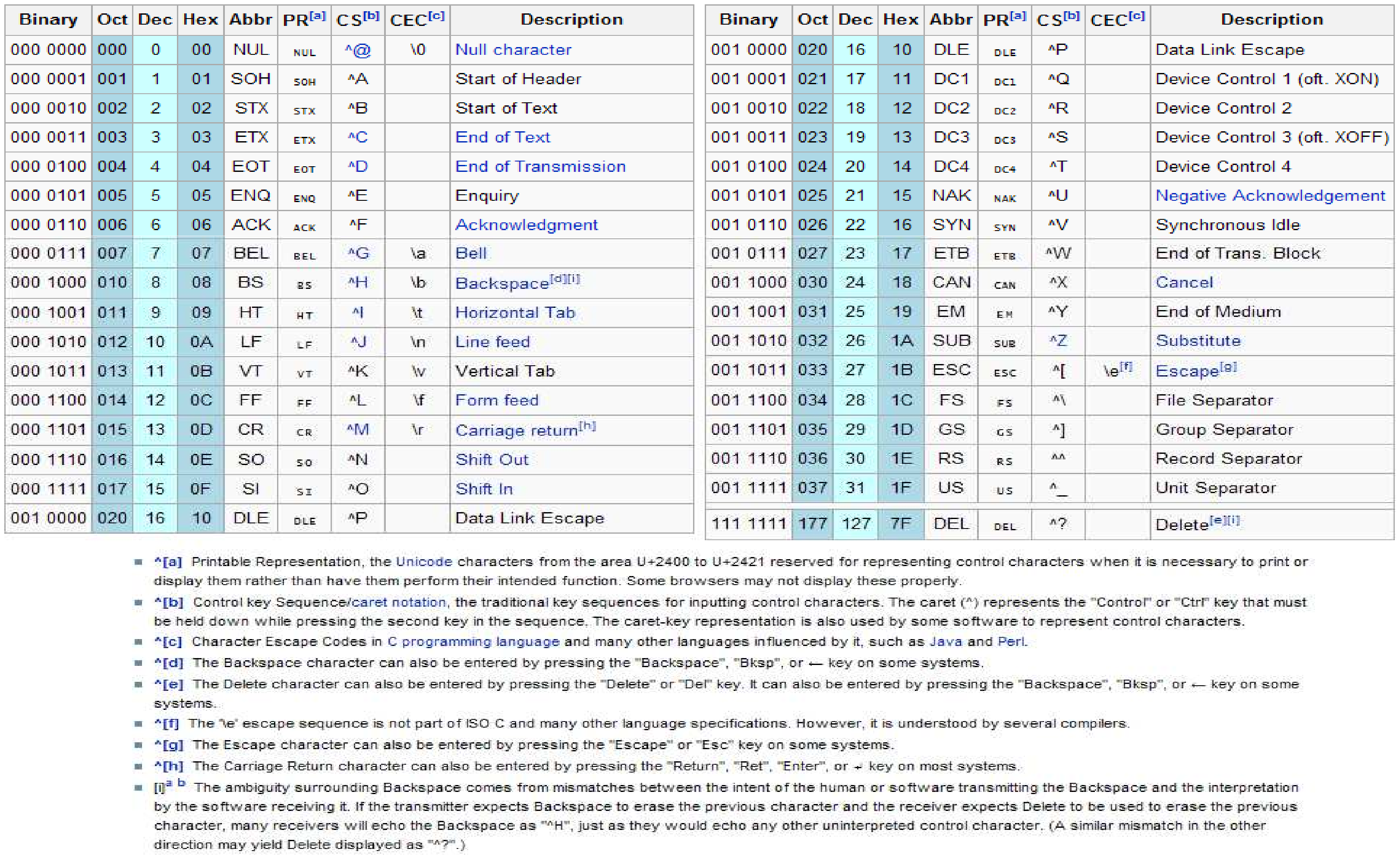
A	·−	I	··	Q	−−·−	Y	−·−−	1	·−−−−
B	−···	J	·−−−	R	·−·	Z	−−··	2	··−−−
C	−·−·	K	−·−	S	···	Period	·−·−·−	3	···−−
D	−··	L	·−··	T	−	Comma	−−··−−	4	····−
E	·	M	−−	U	··−	?	··−−··	5	·····
F	··−·	N	−·	V	···−	/	−··−·	6	−····
G	−−·	O	−−−	W	·−−	@	·−−·−·	7	−−···
H	····	P	·−−·	X	−··−			8	−−−··
								9	−−−−·
								0	−−−−−

Character codes: ASCII

American Standard Code for Information Interchange: control characters

7-bit binary code with fixed code word length for teletypes

(Non-printable commands for teletype printers)



Binary	Oct	Dec	Hex	Abbr	PR[a]	CS[b]	CEC[c]	Description
000 0000	000	0	00	NUL	NUL	^@	\0	Null character
000 0001	001	1	01	SOH	SOH	^A		Start of Header
000 0010	002	2	02	STX	STX	^B		Start of Text
000 0011	003	3	03	ETX	ETX	^C		End of Text
000 0100	004	4	04	EOT	EOT	^D		End of Transmission
000 0101	005	5	05	ENQ	ENQ	^E		Enquiry
000 0110	006	6	06	ACK	ACK	^F		Acknowledgment
000 0111	007	7	07	BEL	BEL	^G	\a	Bell
000 1000	010	8	08	BS	BS	^H	\b	Backspace[d][i]
000 1001	011	9	09	HT	HT	^I	\t	Horizontal Tab
000 1010	012	10	0A	LF	LF	^J	\n	Line feed
000 1011	013	11	0B	VT	VT	^K	\v	Vertical Tab
000 1100	014	12	0C	FF	FF	^L	\f	Form feed
000 1101	015	13	0D	CR	CR	^M	\r	Carriage return[h]
000 1110	016	14	0E	SO	SO	^N		Shift Out
000 1111	017	15	0F	SI	SI	^O		Shift In
001 0000	020	16	10	DLE	DLE	^P		Data Link Escape

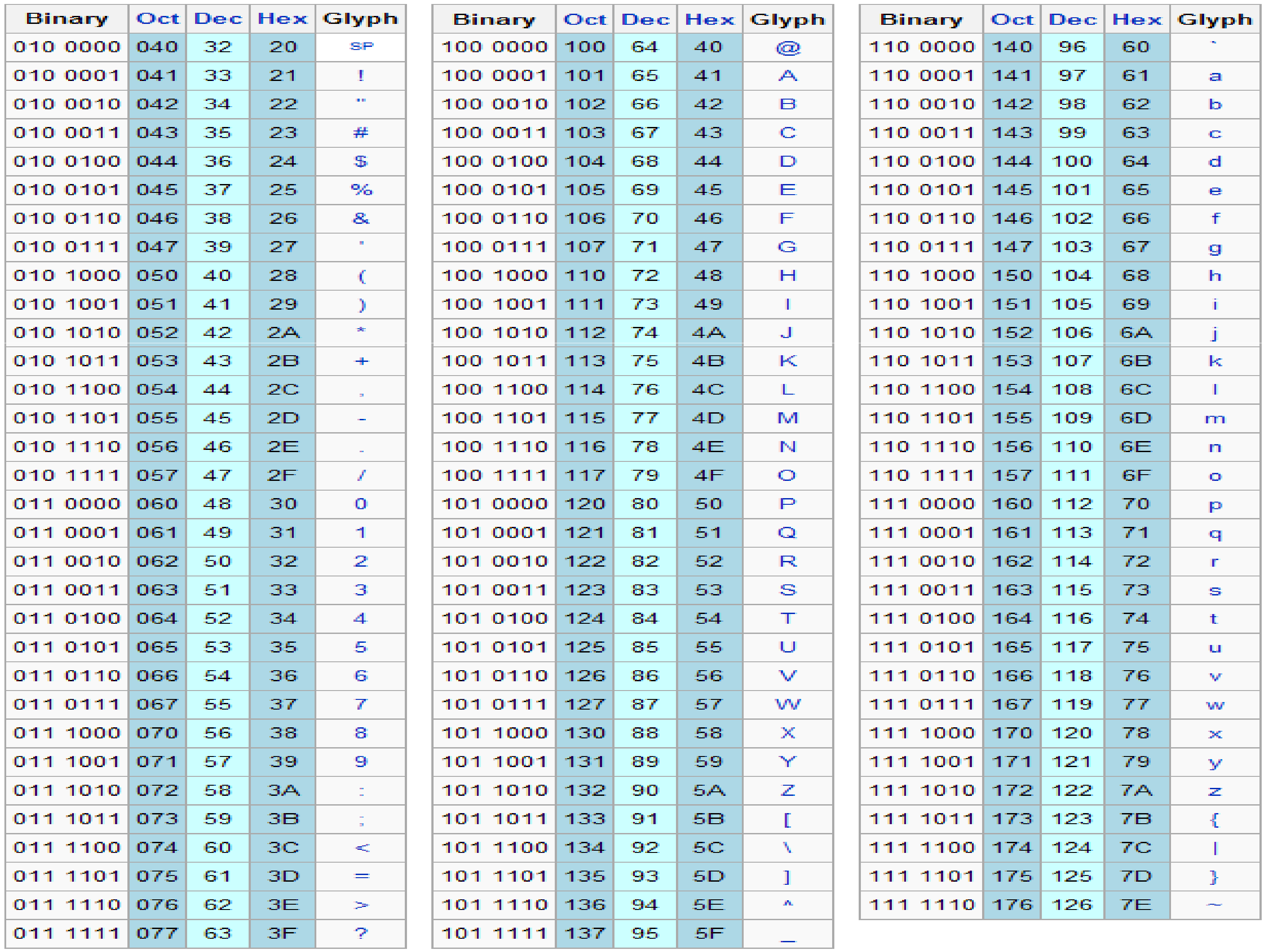
Binary	Oct	Dec	Hex	Abbr	PR[a]	CS[b]	CEC[c]	Description
001 0000	020	16	10	DLE	DLE	^P		Data Link Escape
001 0001	021	17	11	DC1	DC1	^Q		Device Control 1 (oft. XON)
001 0010	022	18	12	DC2	DC2	^R		Device Control 2
001 0011	023	19	13	DC3	DC3	^S		Device Control 3 (oft. XOFF)
001 0100	024	20	14	DC4	DC4	^T		Device Control 4
001 0101	025	21	15	NAK	NAK	^U		Negative Acknowledgement
001 0110	026	22	16	SYN	SYN	^V		Synchronous Idle
001 0111	027	23	17	ETB	ETB	^W		End of Trans. Block
001 1000	030	24	18	CAN	CAN	^X		Cancel
001 1001	031	25	19	EM	EM	^Y		End of Medium
001 1010	032	26	1A	SUB	SUB	^Z		Substitute
001 1011	033	27	1B	ESC	ESC	^[	\e[f]	Escape[g]
001 1100	034	28	1C	FS	FS	^\		File Separator
001 1101	035	29	1D	GS	GS	^]		Group Separator
001 1110	036	30	1E	RS	RS	^^		Record Separator
001 1111	037	31	1F	US	US	^_		Unit Separator
111 1111	177	127	7F	DEL	DEL	^?		Delete[e][i]

- ^[a] Printable Representation, the Unicode characters from the area U+2400 to U+2421 reserved for representing control characters when it is necessary to print or display them rather than have them perform their intended function. Some browsers may not display these properly.
- ^[b] Control key Sequence/caret notation, the traditional key sequences for inputting control characters. The caret (^) represents the "Control" or "Ctrl" key that must be held down while pressing the second key in the sequence. The caret-key representation is also used by some software to represent control characters.
- ^[c] Character Escape Codes in C programming language and many other languages influenced by it, such as Java and Perl.
- ^[d] The Backspace character can also be entered by pressing the "Backspace", "Bksp", or ← key on some systems.
- ^[e] The Delete character can also be entered by pressing the "Delete" or "Del" key. It can also be entered by pressing the "Backspace", "Bksp", or ← key on some systems.
- ^[f] The '\e' escape sequence is not part of ISO C and many other language specifications. However, it is understood by several compilers.
- ^[g] The Escape character can also be entered by pressing the "Escape" or "Esc" key on some systems.
- ^[h] The Carriage Return character can also be entered by pressing the "Return", "Ret", "Enter", or ↵ key on most systems.
- [i]a b The ambiguity surrounding Backspace comes from mismatches between the intent of the human or software transmitting the Backspace and the interpretation by the software receiving it. If the transmitter expects Backspace to erase the previous character and the receiver expects Delete to be used to erase the previous character, many receivers will echo the Backspace as "^H", just as they would echo any other uninterpreted control character. (A similar mismatch in the other direction may yield Delete displayed as "^?".)

See: http://en.wikipedia.org/wiki/Ascii, Download 30.11.2008

Character codes: ASCII

American Standard Code for Information Interchange: printable characters



Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	SP
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^
101 1111	137	95	5F	_
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

See: http://en.wikipedia.org/wiki/Ascii, Download 30.11.2008

Character codes: ASCII

American Standard Code for Information Interchange: binary interpretation

Convert lower case characters into upper case:

upper = lower - (‚a‘ - ‚A‘)
upper = lower - ($110\ 0001_{bin}$ - $100\ 0001_{bin}$)
upper = lower - (97_{dez} - 65_{dez})
upper = lower - 32_{dez}

e.g. upper = ‚s‘ - 32_{dez}
upper = $111\ 0011_{bin}$ - $010\ 0000_{bin}$
upper = $101\ 0011_{bin}$
upper = ‚S‘

Convert upper case characters into lower case:

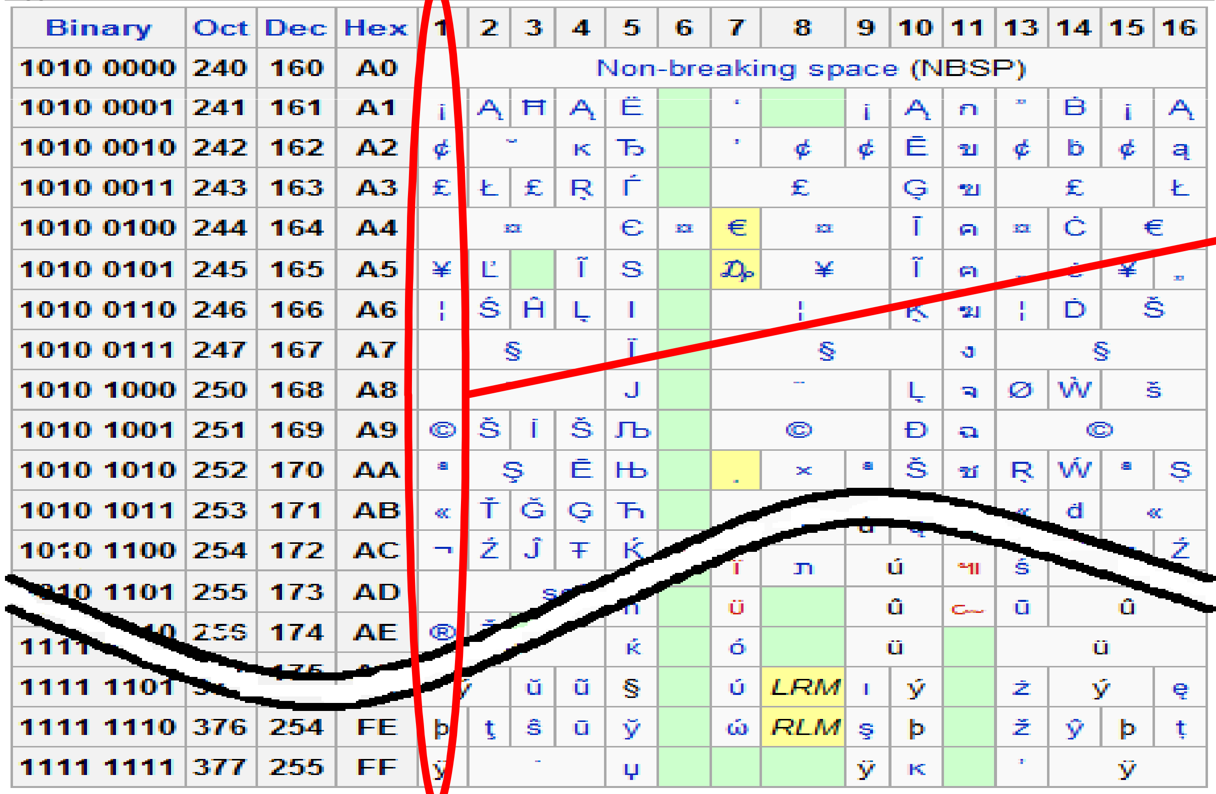
lower = upper + (‚a‘ - ‚A‘)

Character codes: “ANSI” (ISO 8859)

American Standard Code for Information Interchange: control characters

8-bit binary code with fixed code word length for teletypes as extension to the ASCII-code

0000 0000 … 0111 1111	ASCII
1000 0000 … 1001 1111	unused control characters



Binary	Oct	Dec	Hex	1	2	3	4	5	6	7	8	9	10	11	13	14	15	16
1010 0000	240	160	A0		Non-breaking space (NBSP)													
1010 0001	241	161	A1	¡	Ą	Ħ	Ą	Ë		ʽ		¡	Ą	ก	”	Ḃ	¡	Ą
1010 0010	242	162	A2	¢	˘		ĸ	Ђ		ʼ	¢	¢	Ē	ข	¢	ḃ	¢	ą
1010 0011	243	163	A3	£	Ł	£	Ŗ	Ѓ		£			Ģ	ฃ	£			Ł
1010 0100	244	164	A4	¤				Є	¤	€	¤		Ī	ค	¤	Ċ	€	
1010 0101	245	165	A5	¥	Ľ		Ĩ	Ѕ		₯	¥		Ĩ	ฅ	„	ċ	¥	„
1010 0110	246	166	A6	¦	Ś	Ĥ	Ļ	І		¦			Ķ	ฆ	¦	Ḋ	Š	
1010 0111	247	167	A7	§				Ї		§				ง	§			
1010 1000	250	168	A8					Ј		¨			Ļ	จ	Ø	Ẁ	š	
1010 1001	251	169	A9	©	Š	İ	Š	Љ		©			Đ	ฉ	©			
1010 1010	252	170	AA	ª	Ş		Ē	Њ			×	ª	Š	ช	Ŗ	Ẃ	ª	Ș
1010 1011	253	171	AB	«	Ť	Ğ	Ģ	Ћ		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	«	ḋ	«	
1010 1100	254	172	AC	¬	Ź	Ĵ	Ŧ	Ќ	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Ź
[illegible]10 1101	255	173	AD		[illegible]			[illegible]		ï	ת	ú		ฯ	ś	[illegible]		
[illegible]	256	174	AE	®	[illegible]	[illegible]	[illegible]	ћ		ü		û		~	ū	û		
1111 [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ќ		ó		ü			ü			
1111 1101	[illegible]	[illegible]	[illegible]	ý		ŭ	ũ	§		ú	LRM	ı	ý		ż	ý		ę
1111 1110	376	254	FE	þ	ţ	ŝ	ū	ў		ώ	RLM	ş	þ		ž	ŷ	þ	ț
1111 1111	377	255	FF	ÿ	˙			џ				ÿ	ĸ		’	ÿ		

Latin-1: Western European

See: http://en.wikipedia.org/wiki/ISO_8859, Download 30.11.2008

Character codes: UTF-8

Unicode Transformation Format

binary code with variable code word length for internet communication

UCS-4 range (hex.)	UTF-8 octet sequence (binary)
0000 0000 – 0000 007F	0xxxxxxx (ASCII)
0000 0080 – 0000 07FF	110xxxxx 10xxxxxx
0000 0800 – 0000 FFFF	1110xxxx 10xxxxxx 10xxxxxx
0001 0000 – 001F FFFF	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx
0020 0000 – 03FF FFFF	111110xx 10xxxxxx ... 10xxxxxx
0400 0000 – 7FFF FFFF	1111110x 10xxxxxx ... 10xxxxxx

$\underbrace{\texttt{1111110x}}_{\text{Startbyte Number of bytes}}\ \underbrace{\texttt{10xxxxxx ... 10xxxxxx}}_{\text{Coded characters}}$

e.g.

€	U+20AC	00100000 10101100	11100010 10000010 10101100	0xE2 0x82 0xAC

See: http://tools.ietf.org/html/rfc2279, Download 30.11.2008
See: http://www.cl.cam.ac.uk/~mgk25/unicode.html , Download 30.11.2008
See: http://de.wikipedia.org/wiki/Utf-8 , Download 30.11.2008

Character codes: others

Binary-coded decimal (BCD)

Extended Binary Coded Decimal Interchange Code (EBCDIC)

Unicode transformation formats (UTF-7, UTF-16, UTF-32)

…

5 901234 123457

Barcodes

Reference:
http://de.wikipedia.org/wiki/EAN,
Download 30.11.2008

Character codes

What is the difference between ASCII and binary representation of a number?

Remember ASCII-Code: Write a Matlab-program converting the ASCII-representation of „246“ (a=`246`;) into an equivalent decimal output 246 (ans=246;) without using a maybe given Matlab-function doing that for you.

Remember ASCII-Code: How can you easily convert lower-case letters to upper-case one without using given functions like „toupper“ in C? Give the schematic steps!

Matlab (I)

Thank you!