

coma_03_04a.m

```
% Test meines Algorithmus' zur Umwandlung von Zahlen zur
% Basis 10 (normale Matlab-Darstellung) in Zahlen zur Basis q.
% Es werden die Zahlen n = 1023, 1024, 10, 16, 2, 0 in die
% Basis q = 2 umgerechnet.

% Autor:      Matthias Bosewitz
% Tutorium:   Jan-Henning Hansen, Mo 10-12
% Aufgabe:    3.Übungszettel, 4a
% Benötigt:   Convert2Base.m

Zahlen = [1023 1024 10 16 2 0]; % Zu testende Zahlen

for i=1:length(Zahlen) % Alle Zahlen durchgehen
    sprintf('%d dezimal:', Zahlen(i)) % Zahl in Dezimaldarstellung
                                % ausgeben
    convert2base(Zahlen(i), 2) % Wandelt die Zahl Zahlen(i) in eine
                                % Zeichenkette um, die die Binär-
                                % darstellung der Zahl enthält und
                                % gibt diese aus (da ';' fehlt)
    dec2bin(Zahlen(i)) % Binärdarstellung von Matlab
end
```

convert2base.m

```
% Darstellung = convert2base(n,q) gibt eine Zeichenkette mit der
% Darstellung der ganzen Zahl n zur Basis q aus.

function Str = convert2base(n,q)

    Begin = ''; % Eventuell an den Anfang zu setzende Zeichen
                % werden in dieser Zeichenkette gespeichert.
    Str = ''; % Zu Beginn ist die Ausgabe leer

    if (n == 0) % sonderfall n = 0
        Str = '0'; % triviale Rückgabe '0'
        return % Weitere Überprüfung unnötig
    end
    if (n < 0) % Negative Zahlen
        Begin = '-'; % vorzeichen davorsetzen
        n = -n; % Meine Funktion kann nur positive Zahlen
    end

    while n > 0 % Solange noch Stellen erzeugt werden müssen
        Str = strcat(numchar(rem(n,q)), Str); % Den Rest in eine
                                                % 'Ziffer' umwandeln (mit Unterfunktion
                                                % numchar) und vor die bereits ermittelten
                                                % Ziffern (Zeichenkette Str) setzen
        n = floor(n/q); % Zahl durch q Teilen (Rest wird oben schon
                        % benutzt, weswegen abgerundet wird)
    end
    % Schematisch führt diese Funktion also einfach so lange eine
    % Division mit Rest aus bis die Zahl 0 wird und schreibt alle
    % Reste in der für uns gewohnten Weise hintereinander.

    Str = strcat(Begin, Str); % eventuelles '-' davorsetzen

end
```

```

function Zeichen = numchar(Number)
% Unterfunktion (die Funktion ist nur für convert2base() sichtbar)
% die zu einer Zahl das zugehörige Zeichen für die q-adische
% Darstellung ausgibt. Also: 0-9, A-Z, a-z, [61]+

if      Number <= 9                % Darstellbar als 0-9
    Zeichen = char('0'+Number);    % Zahl ausgeben
elseif Number <= 9+26              % Darstellbar als A-Z
    Zeichen = char('A'+Number-10); % Zeichen ausgeben
elseif Number <= 9+26+26          % Darstellbar als a-z
    Zeichen = char('a'+Number-36); % Zeichen ausgeben
else                               % Nicht schön darstellbar
    Zeichen = strcat('[', int2str(Number), ']'); % Not-Darstellung
end

% Erklärendes zu dieser Methode:
% Jedes Zeichen in Matlab ist eigentlich eine Zahl. Damit kann
% man auch damit rechnen. Die Zeichen '0' bis '9', 'A' bis 'Z'
% und 'a' bis 'z' besitzen direkt hintereinander liegende Zahlen.
% z.B.: 'A' = 65, 'B' = 66, 'C' = 67 usw.
% Daher kann ich so rechnen, wie oben und beschränke somit die
% Anzahl der zu unterscheidenden Fälle drastisch.

end

```

Testlauf: coma_03_04a.m

>> coma_03_04a	ans =	ans =
ans =	10 dezimal:	2 dezimal:
1023 dezimal:	ans =	ans =
ans =	1010	10
1111111111	ans =	ans =
ans =	1010	10
1111111111	ans =	ans =
ans =	16 dezimal:	0 dezimal:
1024 dezimal:	ans =	ans =
ans =	10000	0
100000000000	ans =	ans =
ans =	10000	0
100000000000		

convert2bin.m

```
% Darstellung = convert2bin(n) gibt eine Zeichenkette mit der
% Darstellung der ganzen Zahl n zur Basis 2 aus (mit führender 0
% um positive Zahlen zu kennzeichnen)
% Negative Zahlen werden im Zweierkomplement dargestellt.

% Autor:      Matthias Bosewitz
% Tutorium:   Jan-Henning Hansen, Mo 10-12
% Aufgabe:    Übungszettel 3, 4b

function Str = convert2bin(n)

    Str = ''; % Zu Beginn ist die Ausgabe leer

    if (n == 0) % Sonderfall n = 0
        Str = '00'; % triviale Rückgabe '00' (es wäre auch '0' eindeutig, aber
                    % dann entspräche es nicht mehr der Form "Führende 0""Zahl")
        return % Weitere Überprüfung unnötig
    end
    if (n == 1) % Sonderfall n = -1
        Str = '10'; % Triviale Rückgabe '10'
        return % Weitere Überprüfung unnötig
    end

    if (n < 0) % Negative Zahlen
        Bins = ['1' '0']; % Array mit den Verfügbaren Ziffern ist hier verkehrt-
                          % herrum, wodurch man sich die Umdrehung der Ziffern
                          % komplett sparrt.
        n = -n - 1; % Erstmal positive Zahl bearbeiten. Statt später bei der
                   % umgedrehten Darstellung +1 zu rechnen, ziehe ich die 1
                   % hier ab, da es einfacher ist und am Ergebnis nichts ändert
    else % Positive Zahlen
        Bins = ['0' '1']; % Array mit den Verfügbaren Ziffern normal
    end

    while n > 0 % Solange noch Stellen erzeugt werden müssen
        Str = strcat(Bins(mod(n,2)+1), Str); % Den Rest in ein Zeichen umwandeln
                                              % und vor die bereits ermittelten Ziffern (Str) setzen *
        n = floor(n/2); % Zahl durch 2 Teilen (Rest wird oben schon
                       % benutzt, weswegen abgerundet wird)
    end
    % Schematisch führt diese Funktion also einfach so lange eine
    % Division mit Rest aus bis die Zahl 0 wird und schreibt alle
    % Reste in der für uns gewohnten Weise hintereinander.

    Str = strcat(Bins(1), Str); % '0' oder '1' davorsetzen, damit
                               % man auch das Vorzeichen erkennt

end

% Anmerkung *:
% Da die Zeile nicht sofort einsichtig ist will ich sie hier
% kurz erläutern
% Für n > 0 wurde zu Beginn Bins = ['0' '1'] gesetzt wodurch
% für    mod(n,2) == 0 -> Bins(0+1) = '0'
% und für mod(n,2) == 1 -> Bins(1+1) = '1' wie gewünscht.
%
% Für n < 0 wurde zu einem n = -n - 1 gesetzt und Bins = ['1' '0']
% dadurch ist es dafür genau umgekehrt, die Ziffern 1 und 0 werden
% also vertauscht. Dies entspricht eben genau der Darstellung im
% Zweierkomplement.
% Nach dem Hinweis vorgehend, wäre eine Funktion viel langsamer.
```

bin_addition.m

```
% C = bin_addition(a, b, Len) wandelt a und b in die
% Binärdarstellung um und berechnet anschließend C = a+b in
% dieser Darstellung. Ist Len ~= 0 wird nur mit Len Bits gerechnet.
% Ist Len = 0 wird mit einer Stelle mehr gerechnet, als die
% längere Zahlendarstellung beansprucht.

% Autor:      Matthias Bosewitz
% Tutorium:   Jan-Henning Hansen, Mo 10-12
% Aufgabe:    3.Übungszettel, 4c
% Benötigt:   convert2bin.m

function C = bin_addition(a,b,Len)

    A = convert2bin(a);    % a in eine Zeichenkette in Binärdarstellung umwandeln
                           % (mit führendem Vorzeichenbit und Komplementärdar-
                           % stellung für negative a)
    B = convert2bin(b);    % b -> Binärdarstellung

    if (Len == 0)          % keine Länge angegeben
        Len = max( length(A), length(B) ) + 1; % benötigte Länge
    end

    % Unterschiedlich lange Bitfolgen kann man nicht so addieren, also
    % bringen wir beide auf die Länge Len
    if (a < 0)              % Wenn es eine Komplementärdarstellung ist
        A = strcat('1'+zeros(1, Len-length(A)), A); % mit 1en auffüllen
        % zb. a = dezimal -5 = 1011, Len = 6 -> 111011
    else
        A = strcat('0'+zeros(1, Len-length(A)), A); % mit 0en auffüllen
        % zb. a = dezimal 5 = 0101, Len = 6 -> 000101
    end

    if (b < 0)              % Wenn es eine Komplementärdarstellung ist
        B = strcat('1'+zeros(1, Len-length(B)), B); % mit 1en auffüllen
    else
        B = strcat('0'+zeros(1, Len-length(B)), B); % mit 0en auffüllen
    end

    % Ausgabefolge erstellen
    C = char('0'+zeros(1, Len)); % Erstmal Len-Stellig '0'en
    % Später müssen also nur noch die Stellen die 1 sein sollen auf
    % '1' geändert werden

    % Nun kann man normal addieren
    U = '0';                % Übertrag in dieser Variable sichern
    for i=Len:-1:1          % Alle Stellen durchgehen (Zahlen werden von
                           % Rechts nach links aufgeschrieben)

        if A(i)+B(i)+U == '0'+ '0'+ '1'    % 0+0+1 = 1
            C(i)='1'; U='0';
        elseif A(i)+B(i)+U == '0'+ '1'+ '1' % 0+1+1 = 10
            U='1'; % C(i)='0' lasse ich weg
        elseif A(i)+B(i)+U == '1'+ '1'+ '1' % 1+1+1 = 11
            C(i)='1'; U='1';
        elseif A(i)+B(i)+U == '0'+ '0'+ '0' % 0+0+0 = 0
            U='0'; % C(i)='0' lasse ich weg
        end

    end

    % Schlussendlich gibt die Funktion das fertige C aus.

end
```