

Angewandte Programmierung in der Computerlinguistik - Internationalisierung / Lokalisierung

Andreas Neumann M.A.

Internationalisierung / Lokalisierung

- * Auch Programmcode ist kulturell geprägt
- * Computerprogramme entstehen in einem kulturellen Umfeld und werden innerhalb eines kulturellen Kontexts ausgeführt
- * Es besteht der Anspruch Computerprogramme an den kulturellen Kontext anzupassen in dem sie ausgeführt werden.

Internationalisierung / i18n

- ✳ Bei der Planung und Erstellung des Programms, wird darauf geachtet, dass das Programm einfach lokalisierbar ist
- ✳ z.B. Konfigurierbare Oberfläche, Fähigkeit versch. Schriftsysteme zu verwenden ...

Lokalisierung

- * Anpassung eines Programmes an einen (kulturellen/speziellen) Kontext
- * Kann u.a. Sprache, Konventionen, GUI usw. betreffen
- * Wird für jeden Kontext in dem ein Programm ausgeführt werden soll, von neuem durchgeführt

Lokalisierung: Schrift- und Zeichensystem

- * ASCII
- * ISO-8559-X
- * Unicode

ASCII

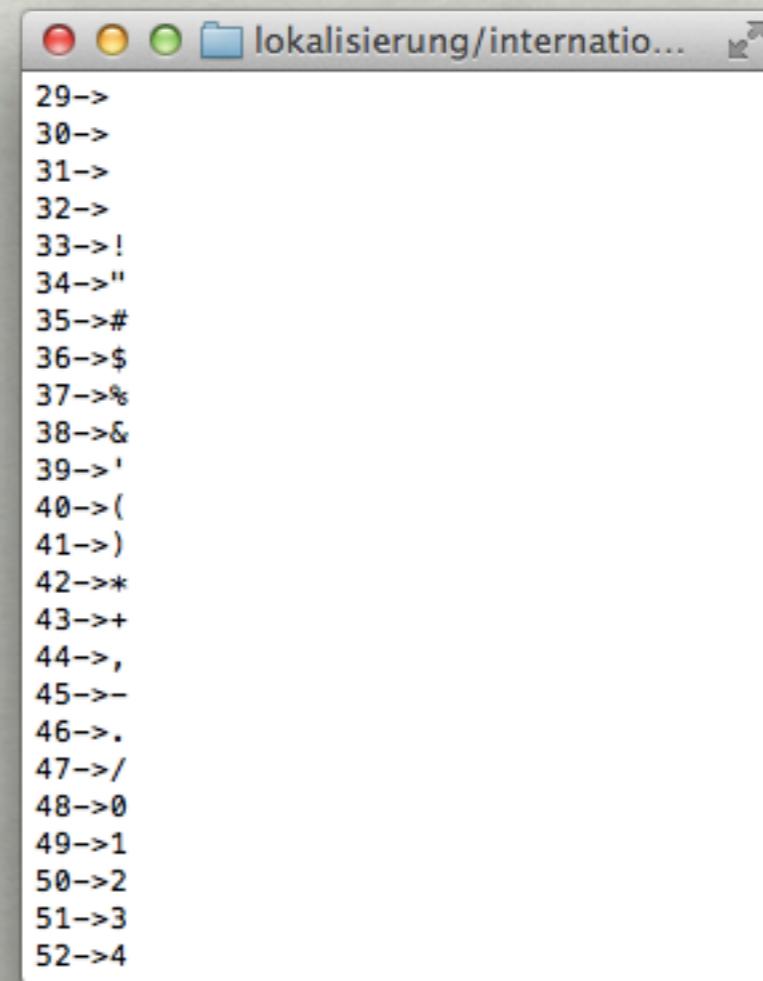
- * 128 - Zeichen
- * 7 - bit
- * englisch/amerikanischer Kulturraum

ASCII - Perl

```
#!/usr/bin/perl -w
# Gibt alle Zeichen im ASCII Bereich aus

@ascii = (0..127);

foreach(@ascii) {
    print($_ . "->" . chr($_) . "\n");
};
```



The screenshot shows a terminal window titled "lokalisierung/internatio...". The window displays a list of ASCII character codes and their corresponding symbols or escape sequences. The list starts at code 29 and ends at code 52. The output is as follows:

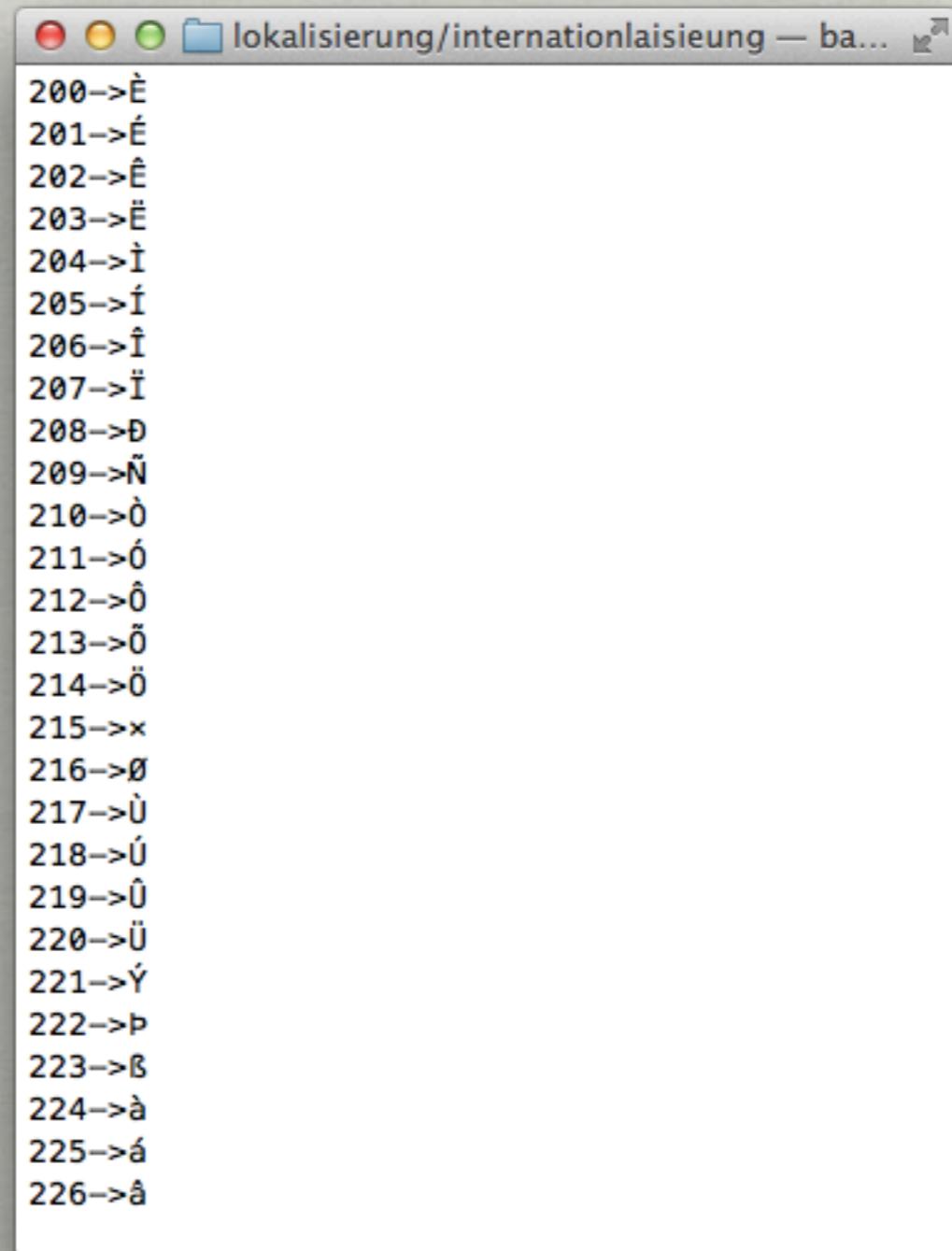
```
29->
30->
31->
32->
33->!
34->""
35->#
36->$
37->%
38->&
39->'
40->(
41->)
42->*
43->+
44->,
45->-
46->.
47->/
48->0
49->1
50->2
51->3
52->4
```

ISO-8559-X

- * 8bit
- * 256 Zeichen - 128 Zeichen
- * Landestypisch
- * Einer Datei ist nicht anzusehen welche ISO-8559-X Codepage verwendet wird

ISO-8559-1/Latin1

```
#!/usr/bin/perl -w
foreach(0..255) {
    print("$_->".chr($_)."\n");
}
```

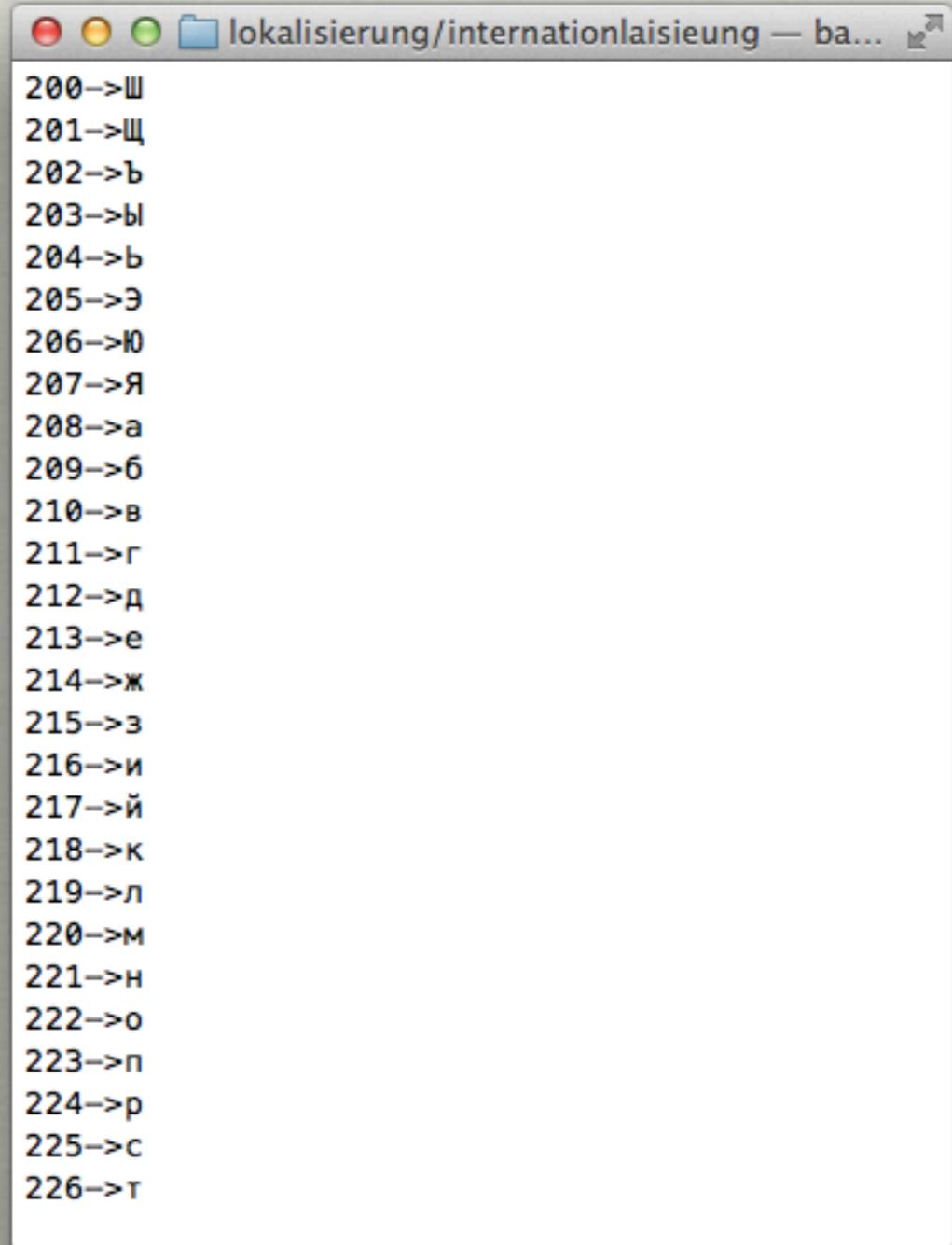


The screenshot shows a terminal window titled "lokalisierung/internationlaisierung — ba...". The window displays a list of character mappings from their ASCII codes to their corresponding ISO-8559-1 characters. The output is as follows:

```
200->È
201->É
202->Ê
203->Ë
204->Ì
205->Í
206->Î
207->Ï
208->Ð
209->Ñ
210->Ò
211->Ó
212->Ô
213->Õ
214->Ö
215->x
216->Ø
217->Ù
218->Ú
219->Û
220->Ü
221->Ý
222->Þ
223->ß
224->à
225->á
226->â
```

ISO-8559-5 / Kyrillisch

```
#!/usr/bin/perl -w
foreach(0..255) {
    print("$_->".chr($_)."\n");
}
```



The screenshot shows a terminal window titled "lokalisierung/internationlaisierung — ba...". The window displays a list of character mappings from their ASCII codes to their corresponding Cyrillic characters. The output is as follows:

```
200->Ш
201->Щ
202->ѣ
203->ы
204->ѣ
205->Э
206->Ю
207->Я
208->а
209->б
210->в
211->г
212->д
213->е
214->ж
215->з
216->и
217->й
218->к
219->л
220->м
221->н
222->օ
223->ո
224->ր
225->ս
226->տ
```

ISO-8559-X - Schwächen

- * Schriftsysteme können nicht gemischt werden
- * Nicht geeignet für Sprachen mit mehr als 128 Zeichen, wie z.B. Chinesisch
- * unvorhersehbar, da Schriftsystem nicht aus dem binären Abbild der Datei zu erkennen ist

Unicode



- * UCS 2 / UTF-16
- * UCS 4 / UTF-32
- * UTF-8

Warum Unicode ?

- ✳ Notwendigkeit verschiedene Schriftsystem zu unterstützen
- ✳ Schriftsystem mit mehr als 256 Zeichen unterstützen
- ✳ Gleichzeitige Darstellung verschiedner Schriftsysteme nebeneinander
- ✳ Vereinheitlichung: Unicode-Text ist nicht von der Locale abhängig

UCS 2 / 4 - UTF-16/UTF-32

- * 2 Byte bzw. 4 Byte pro Zeichen
- * ergo 8 bit bzw. 16 bit
- * UCS 2 / UTF-16 Windows und ursprünglich Java
- * UCS 4 / UTF-32 Viele Sprachen intern / Linux

Unicode

- * aktuelles zu den Codeblöcken: [http://
de.wikipedia.org/wiki/Liste_der_Unicodebl
%C3%B6cke](http://de.wikipedia.org/wiki/Liste_der_Unicodebl%C3%BCcke)

Unicode Schwächen

- * zwei bis vierfacher Speicherbedarf

UTF-8

- * alternative Kodierung mit gleicher Mächtigkeit wie UTF32
- * Versucht Problem des Speicherverbrauchs zu lösen
- * variable Bytelängen pro Character (1-4 Byte)

UTF-8 - Stäreken

- * geringerer Speicherbedarf als UTF-32
- * Sortierbarkeit bleibt erhalten
- * Rückwärtskompatibilität zu ASCII

UTF-8 - Schwächen

- * Gleichförmigkeit geht verloren
- * komplexere Abarbeitung!

Komponenten einer Programmiersprache, die bei Internationalisierung betroffen sind

- * Aktivierung der Unterstützung
- * Internationalisierung von Strings
- * Internationalisierung des Programmcodes
- * IO
- * Methoden der Programmiersprache

UTF-8 bei Perl aktivieren

- * Beim Aufruf: **perl -C**
- * Mit Pragma: **use utf8;**

Strings

- * minimal: Unicode-Bytefolgen dürfen in einen String gespeichert werden
- * optimal: Unicode-Zeichen werden korrekt als Character gespeichert

Unicode Strings - Beispiel in Perl

```
#!/usr/bin/perl -w

use utf8;

@cities = ("Москва", "北京", "नई दिल्ली");

foreach my $city (@cities) {
    print "City: $city ->".length($city)."\n";

    foreach my $char (split//,$city) {
        print $char.":".ord($char)."\n";
    }
}
```

Programmcode

- * Unicode Zeichen als Bezeichner für Variablen und Funktionsnamen erlaubt

Unicode im Programmcode

- Beispiel in Perl

```
#/usr/bin/perl -w
use utf8;

$Москвá = "Moskau";
$beijing = "北京";

λόγος($Москвá);

sub λόγος {
    my $in = shift;
    print("$in\n");
}
```

Internationalsierung IO

- * Können verschiedene UNICODE-Kodierungen gelesen und geschrieben werden
- * Kann ein Programm über STDIN, STDOUT und STDERR mit Unicode-Kodierungen umgehen

IO Filehandle - Beispiel in Perl

```
#!/usr/bin/perl -w

#Author: Andreas Neumann
#Date: 17.11.2007
#I/O with UTF-8

use open ":utf8"; #For all I/O operations except STDIN, STDOUT und
STDERR

# FILES
open (F, "<", $ARGV[0]);
open (OUT, ">:utf8", "output.txt"); #just for the Filehandle

while (<F>) { print OUT; }

close F;
close OUT;
```

IO Standardkanäle - Beispiel in Perl

```
#!/usr/bin/perl -w

#Author: Andreas Neumann
#Date: 18.11.2007
#Synopsis: STD I/O with UTF-8

binmode(STDIN,:utf8);
binmode(STDOUT,:utf8);
binmode(STDERR,:utf8);

while (<>) {
    print length;
    print;
}
```

Methoden der Programmiersprache

- * Arbeiten alle Methoden auf Characterbasis korrekt mit UTF-8
- * Typische Beispiele: *chop()*, *chomp()*, *substr()*, *pos()*, *index()*, *rindex()*, *sprintf()*, *write()*, and *length()*.

Perl Methoden und UTF-8 Kompatibilität

- * mit dem Pragma **use utf8** arbeiten die meisten Perl Funktionen korrekt mit utf8
- * Ausnahmen sind Systemnahe Funktionen, wie z.B. pack und unpack

Perl Methoden und UTF-8 - Beispiel *length()*

- * Nur mit use utf8; werden die Methoden umgestellt

```
andi$ perl -e "print length(\"Bär\")."\n;"  
4  
andi$ perl -e "use utf8; print length(\"Bär\")."\n  
\";"  
3
```

Perl Methoden und UTF-8 - Beispiel *split()*

- * Ohne use utf8

```
andi$ perl -e"print join('-',split('//,\\"Bär\\")).\"\\n\";"  
B-?-?-r
```

- * mit pragma **use utf8;**

```
andi$ perl -e"use utf8;print join('-',split('//,\\"Bär\\")).\"\\n\";"  
B-?-r
```

- * mit angepasstem IO

```
andi$ perl -CIOE -e"use utf8;print join('-',split('//,\\"Bär\\")).\"\\n\";"  
B-ä-r
```

Reguläre Ausdrücke und Unicode-Kodierungen

- * Unicode hat weitreichende Auswirkungen auf das Arbeiten regulärer Ausdrücke
- * z.B. Was bezeichnet nun \w
- * Unicode *pragmas* \p{}
- * Für litarale Matches ist use utf8 notwendig

```
#!/usr/bin/perl -X
use utf8;

open (UNITXT,"<:utf8","unicode.txt");

while (<UNITXT>) {

#Script Arabic / Alle arabischen Wörter finden
    while ($_=~/(\p{Arabic}+)/g) {
        $ar++;
        push(@ar,$1);
    }
#Property punctuation / Satzzeichen finden
    while (/(\p{Punctuation})+/g) {
        $punc++;
        push(@punc,$1);
    }
#Uppercase / Alle Großbuchstaben finden
    while (/(\p{Uppercase_Letter})/g) {
        $big++;
        push(@big,$1);
    }

    if (/،/bmiyin/) {
        print;
    }

}
```

```
$ perl -CIOE uctextexample.pl
```

הירשמו כעת לכנס 12-10 במרץ 1997, במינץ שבגרמניה. בכנס השתתפו מומחים מכל ענפי התעשייה בנושא האינטרנט העולמי במערכות הפעלה ובישומים, בגופנים, בפריסת טקסט ובסוחר רבלשיוני.

Die Datei enthält 45 Wörter die aus Buchstaben des arabischen Blocks geformt sind:

أوروبا برمجيات الحاسوب انترنت تصبح عالميا مع يونيكود تسج ل الآن لحضور المؤتمر الدولي العاشر ليونيكود الذي سيعقد في مدينة ماينتس ألمانيا وسيجمع والمحلية على حد سواء مناقشة سبل استخدام يونيكود والمحسبة متعددة اللغات عندما يريد العالم أن يتكل م فهو يتحد ث بلغة يونيكود

Die Datei enthält 266 Punktationszeichen:

```
[ ] : ( ) : / . - . . //:  
/ / / . . //: : : / . . . . . . . . . . . . . . . . . . [ ]  
, , . , - : , . , . . , , : , , , , , , - , : , , , , , . . - . , . : , . -  
. . . : , , . , - , , - , , , , , , - : : - , . -  
( ), ( ) . . . . . . . . . . . . . . : . . , , , , , -  
, - - : , ( , . . . . . . . ) ( ) ( : , ( ) ! , . . , , , : . . -  
,( , , ) ,( , , ) ,( , , ) / / . . //: : - , . , , - , , - , , . , -  
- # - / / . . //: : / / . . //: .( , , )  
. , , ; . ; . ; . ; . ; . ; . ; . ; . ; . ; . ; . ; . ; . ; . ; -
```

Die Datei enthält 141 Großbuchstaben:

```
T T U K M S R L P N V T F F A A E U T S Q G E S I I U M S I U K Z M M D K T B B I  
U I L I U B P S T C W W U R E П И И З Д М К У М Г К И И У К У А H U Y K J U U O S  
U U C S U M U C T U M U G U U U U U S S S G K P T Z M D D L O H R R T A N F Ú P Ó R  
R K H K N Í Á F S T B L M L Ý
```

Unicode - Common Pitfalls

- ✳ Einige Überraschungen:
- ✳ Compound Characters
- ✳ Homographen

Compound Characters

```
#!/usr/bin/perl -w

use utf8;

@chars=("ö","ö");

foreach (@chars) {
    print "$_ -> length:".length($_)." codepoint:".ord($_)."\n";
}

foreach (@chars) {
    foreach(split(//,$_)){
        print "$_ -> length:".length($_)." codepoint:".ord($_)."\n";
    }
}
```

ö -> length:1 codepoint:246
ö -> length:2 codepoint:111

ö -> length:1 codepoint:246
o -> length:1 codepoint:111
" -> length:1 codepoint:776

- * Unicode erlaubt es Zeichen zusammenzusetzen
- * Ob es sich, wie im vorangegangen Beispiel um ein „ö“ oder ein „o“ mit darübergestzten Punkten handelt ist mit bloßem Auge nicht zu erkennen
- * Mehr dazu: http://en.wikipedia.org/wiki/Combining_character

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+030x	ˊ	ˊ	^	~	-	-	ˇ	˙	˝	˙	º	˝	ˇ	˙	˝	˝
U+031x	ˇ	˘	‘	’	‘	’	ˇ	ˊ	ˋ	ˊ	ˋ	ˇ	‘	ˊ	ˋ	ˊ
U+032x	-	҃	҄	·	“	◦	,	ጀ	‘	՚	՞	՞	՞	՞	՞	՞
U+033x	~	-	-	=	~	-	-	/	>	□	□	~	x	ſ	=	
U+034x	ˊ	ˊ	ˇ	,	”	”	ጀ	=	”	՚	՚	՚	՚	՚	՚	՚
U+035x	>	‘	՞	x	<	>	>አ	>	*	՞	՞	՞	՞	՞	՞	՞
U+036x	˜	˜	→	a	e	i	o	u	c	d	h	m	r	t	v	x

Homographen

```
#!/usr/bin/perl -w

$as = ["A", "A", "A"];

foreach(@{$as}) {
    print_char_with_codepoint($_);
}

sub print_char_with_codepoint {
    $char = shift;
    print "char:$char->." Codepoint:".ord($char)."\n";
}
```

```
dhcp27:lokalisierung:internationlaisierung andi$ perl
unicode_look_alike2.pl
char:A-> Codepoint:208
char:A-> Codepoint:65
char:A-> Codepoint:206
```

Homographen

- * Unicode ist in Blöcken mit versch. Schriftsystemen aufgebaut
- * Diese Blöcke bilden ein vollständiges Alphabet ab
- * Einige Zeichen aus versch. Alphabeten sehen gleich aus haben aber einen anderen Codepoint und eine andere Bedeutung
- * Nur auf Codepoint-Basis (numerischer Wert) unterscheidbar, nicht optisch