

# Einführung in die Programmiertechnik

## Darstellung von Text

# „Plain Text“

- Abstraktion: Text wird durch eine Folge von Symbolen (Buchstaben, Zahlen, Interpunktionszeichen) dargestellt
  - Verzicht auf Informationen über Schriftart, Schriftgröße, Schriftfarbe, Seitenstruktur
  - Eingeschränkte Unterstützung für Zeilen- und Absatzstruktur (Zeilenumbrüche, Tabulatorzeichen)
- Alternativen:
  - „rich text“: zuzüglich typographischer Informationen
  - strukturierter Text: zuzüglich editorieller/semantischer Informationen (Kapitelstruktur, Querverweise, Auszeichnung von Zitaten, ...)

# ASCII

- American Standard Code for Information Interchange
  - 1963 erstmalig vorgeschlagen, 1968 normiert (ANSI X3.4-1968)
  - später auch (in Varianten) internationaler Standard
    - ISO 646, CCITT International Alphabet #5
    - international reference version, nationale Varianten
      - DIN 66003: @ vs. §, [ vs. Ä, \ vs. Ö, ] vs. Ü, ...
- 7-Bit-Zeichensatz (128 Zeichen)
  - 32(34) Steuerzeichen, 96(94) druckbare Zeichen
  - heute immer in 8 Bit kodiert (MSB ist dann immer 0)
- systematisch strukturiert
  - Ziffern (0..9), Großbuchstaben (A..Z), Kleinbuchstaben (a..z) sind jeweils durchgehend nummeriert (48..57, 65..90, 97..122)
  - Steuerzeichen: 0..31
    - üblich: 7 (BEL), 9 (TAB), 10 (NL), 13 (CR)

# 8-Bit-Zeichensätze

- Oft Erweiterungen von ASCII: weitere druckbare Zeichen im Bereich 128..255
- ISO 8859: weitere Zeichen im Bereich 192..255 für Europa und den Nahen Osten
  - ISO-8859-1: Westeuropa (Latin-1)
  - ISO-8859-2: Mitteleuropa (Latin-2)
  - ISO-8859-3: Südeuropa (Latin-3)
  - ISO-8859-4: Nordeuropa (Latin-4)
  - ISO-8859-5: Kyrillisch
  - ISO-8859-6: Arabisch
  - ISO-8859-7: Griechisch
  - ISO-8859-8: Hebräisch
  - ISO-8859-9: Türkisch (Latin-5; Austausch der isländischen Zeichen mit türkischen)
  - ISO-8859-10: Nordisch (Latin-6; Latin 4 + Inuit, Nicht-Skolt Sami)
  - ISO-8859-11 (1999): Thai
  - ISO-8859-13: Ostseeraum (Latin-7)
  - ISO-8859-14: Keltisch (Latin-8)
  - ISO-8859-15: Westeuropa (Latin-9, Latin-1 ohne Bruchzeichen, plus €, Š, Ž, œ, Ÿ)
  - ISO-8859-16: Europa (Latin-10, Verzicht auf Symbole zugunsten von Buchstaben)

# Andere Zeichensätze

- Viele proprietäre 8-Bit-Zeichensätze:
  - IBM-Codepages (z.B. CP437)
  - Windows-Codepages (z.B. windows-1252)
  - Mac-Zeichensätze (z.B. mac-roman)
  - Großrechner: EBCDIC
- Mehrbytezeichensätze: ein oder zwei Bytes pro Zeichen
  - Chinesisch: Big5 (traditional Chinese), GB-2312 (simplified Chinese)
  - Japanisch: JIS 0208, JIS 0212
  - Koreanisch, Vietnamesisch
- Standards zur Kombination von Kodierungen: ISO-2022
  - z.B. iso-2022-jp (RFC 1554): kombiniert ASCII, JIS X 0208-1978, GB2312-1980, ...

# Unicode

- Vereinigung aller bekannten Schriftsätze
  - die ersten 256 Zeichen sind identisch mit Latin-1
  - fortlaufende Erweiterung um Symbole, die bisher nicht als Text darstellbar waren
  - ursprünglich 16-Bit-Zeichensatz, jetzt 31-Bit-Zeichensatz (aber: Beschränkung auf  $17 \cdot 2^{16}$  Zeichen)
- Aktuelle Version: Unicode 5.2

# Unicode (2)

- Unicode 5.2: ISO/IEC 10646-2003 (einschließlich Amendments 1 bis 6)
  - 107156 graphische Zeichen
  - 140 Formatierungszeichen
  - 65 Steuerzeichen
  - 867169 reservierte Zeichen
  - Zeichensatz wird UCS-4 genannt
    - UCS-2 ist eine Teilmenge mit < 65536 Zeichen

# Unicode-Prinzipien

- Universalität: Ein Zeichensatz für alle Schriftsysteme
- Effizienz: Einfach zu verarbeiten
- Zeichen, nicht Glyphen
- Semantik: Alle Zeichen haben klare Bedeutung
- Reiner Text: Unicode-Zeichen stellen nur Text dar
- Logische Ordnung: Reihenfolge im Speicher folgt der „logischen“ Ordnung
- Vereinheitlichung (*unification*): Doppelte Zeichen in verschiedenen Schriftsystemen werden nur einmal in Unicode aufgenommen
- Dynamische Komponierung: Neue Zeichen können durch Komposition aus bestehenden entstehen
- Zeichenäquivalenz: Vorkomponierte Zeichen sind „äquivalent“ zu dekomponierten Zeichenfolgen
- Konvertierbarkeit: Unicode kann verlustfrei in andere Zeichensätze konvertiert werden (und zurück)

# Unicode-Zeichen

- haben stabilen Code (code point)
  - z.B. U+00DF
- haben stabilen Zeichennamen
  - z.B. LATIN SMALL LETTER SHARP S
- Unicode-Standard enthält Demo-Glyphen
  - z.B. „ß“
- Unicode-Datenbank gibt Zeicheneigenschaften an
  - z.B. „Letter, lower case“ (LI)

# UTF-8

- Kodierung von Unicode in Bytes: mehrere Bytes pro Zeichen
  - Variante 1: jedes Zeichen benötigt mehrere Bytes (UTF-16, UTF-32)
    - Probleme: Rückwärtskompatibilität, Byte Order
  - Variante 2: Jedes Zeichen benötigt eine variable Zahl von Bytes
    - UTF-8: 1 bis 4 Bytes pro Zeichen
- Zeichen <128: 1 Byte (äquivalent zu ASCII)
- Nullbytefrei
- Jede Bytefolge unterliegt bestimmtem Muster, um die Dekodierung zu erleichtern

## UTF-8 (2)

Bitmuster	Anzahl kodierter Zeichen	Bereich kodierter Zeichen (hex)
0xxxxxxx	128	0..7F
110xxxxx 10xxxxxx	2048	80..7FF
1110xxxx 10xxxxxx 10xxxxxx	65536	800..FFFF
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	2097152 (1114112)	10000..10FFFF

# Zeichenketten

- Kodierung von Zeichenfolgen: Verkettung der Kodierungen der einzelnen Zeichen
- Zeichenkette: engl. „character string“ oder einfach „string“
- Zwei Modi: Bytefolgen oder Zeichenfolgen
  - C, C++, (Python): Strings sind Folgen von Bytes; Interpretation der Bytes entsprechend dem Zeichensatz obliegt der Anwendung
  - Java, C#, (Python): Strings sind Folgen von Unicode-Zeichen; jedes Unicodezeichen ist intern durch mehrere Bytes repräsentiert
- Stringlänge: mehrere Repräsentationen
  - Datei: Stringende = Dateiende
  - C, C++: Stringende wird durch Null-Byte angezeigt
  - Pascal, Java, Python, ....: Zusammen mit dem String wird die Länge gespeichert