Grundlagen der Rechner und Programmierung

- Übersicht der Rechner-Hardware, Netzwerke
  - Handbuch: PC-Technik Grundlagen
  - Handbuch: Netzwerke

- Übersicht der Grundtypen der Rechner-Software

- Grundlagen der Programmierung :
  gemeinsame Strukturen der meisten Programmiersprachen
  - Handbuch: Grundlagen der Programmierung

- **http://www.rrzn.uni-hannover.de/buecher.html**

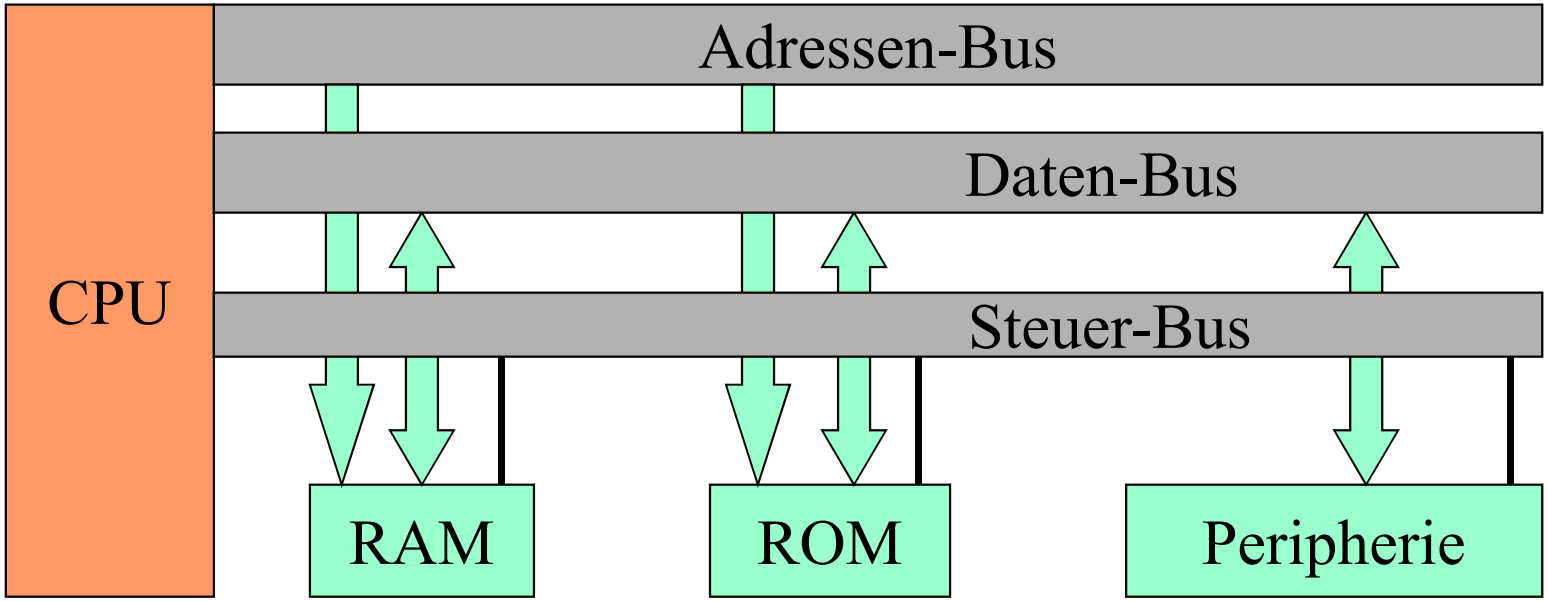
Bestandteile eines digitalen Rechners

- Hardware
  - Prozessor, Speicher, Peripherien

- Software
  - BIOS
  - Betriebssystem
  - Anwendersoftware
    - fertige Anwenderprogramme
    - Eigenbau - „Programmieren“, dafür Compiler (spezielle Anwenderprogramme) notwendig

Hardware

- Grundbestandteile der Hardware:
  - Prozessor (CPU, Abkürzung für Central Processing Unit)
  - Speicher für Programm, für Daten (getrennt/gemeinsam)
  - Ausgabeschnittstellen (Festplatte, VGA, Drucker, Signalleitung)
  - Eingabeschnittstellen (Tastatur, Signalleitung)



Prozessorentypen

- Klassifizierung nach der Registerlänge: 8,16, 32, 64 bit
  - Bedeutung: wie viele Bits werden in einem Schritt gleichzeitig bearbeitet

- Aktuell: 64-Bit-Prozessoren: Intel Core 2 Duo, AMD Athlon 64 X2
  - 45 nm Technologie: Leiterbahnenabstand = 90 nm
  - Leiterbahnbreite etwa 25 nm
  - Vergleich: Ribosome sind etwa 20 nm groß, Si-Gitterkonstante 0,543 nm

- Dual Core, Quad Core: mehrere (2,4) Prozessoren auf einem Chip

- Cache: schneller Zwischenspeicher für Daten/Befehle
  - bis 8 MB je Prozessor (je größer, desto schneller)

RAM – Random Access Memory – flüchtiger Speicher

- Begriffe: Hauptspeicher, RAM, Arbeitsspeicher
  - Der Speicher, mit dem der Prozessor direkt kommunizieren kann und der **in beliebiger Reihenfolge beschreibbar** und **lesbar** ist

- Ausführung: DRAM (Dynamic RAM), SRAM (Static RAM)
  - DRAM: 1 Bit besteht aus 1 Transistor und 1 Kondensator, Refresh nötig
  - SRAM: 1 Bit besteht aus 2 Transistoren, kein Refresh nötig
    - Schneller als DRAM, aber niedrigere Dichte
    - In der CMOS-Ausführung als Einstellungsspeicher im PC

- PC-Speicher: DRAM, Module bis 4 GB (x*Grundtakt 200 MHz)
  - DDR: Double Data Rate, 184-pin (PC3200=DDR400) (x=2)
  - DDR2: 240-pin (PC6400=DDR2-800), 1.8 V (x=4)
  - DDR3: 240-pin (PC12800=DDR3-1600, 1.5 V (x=8)

ROM – Read Only Memory – nichtflüchtiger Speicher

- Haupteigenschaft: der Inhalt bleibt auch nach der Spannungs-abschaltung erhalten
  - Der Speicher, mit dem der Prozessor direkt kommunizieren kann und der **in beliebiger Reihenfolge** nur **lesbar** ist (also auch „random access“)

- Typen
  - ROM – Inhalt bei der Herstellung festgelegt
  - PROM – Inhalt einmal programmierbar
  - EPROM – Erasable PROM – mit UV-Strahlung löschbar, kann wiederholt programmiert werden
  - EEPROM – Electrically Erasable PROM: z.B. USB-Sticks, Speicherkarten

Peripherien

- Schnittstellen: IDE (ATAPI), SCSI, SATA, SATA2
- Serielle und Parallele Schnittstelle, PS/2, USB, FireWire (IEEE-1394)

- Tastatur, Maus, Grafikkarte, Monitor
- Floppy-Disk, Festplatte, CD, DVD/RW
- Speicherkarten, USB-Stick (Solid-State-Memory)

- Netzwerk, Modem, Fax, ISDN
- A/D, D/A-Wandler: Spannungssignal Ein- und Ausgabe
- Sound, Drucker, Scanner, Kamera
- Datensicherung: Streamer, DAT, DLT

PC-Aufbau

- Hauptplatine (Motherboard): mehrschichtige Elektronikplatine mit
  - Spannungsanschluss (+12V, -12V, +5V, -5V, +3.3V)
  - Prozessor
  - Quartz-Taktgeber
  - Sockel für EPROM (BIOS), RAM
  - Echtzeit-Uhr, CMOS-RAM-Chip mit Akku oder Batterie
  - Tastatur- und Mausanschluss
  - Steckplätze (Slots) für Erweiterungskarten (ISA, PCI, AGP, PCIe)
  - Serielle und parallele Schnittstellen
  - Controller (IDE, SATA) für Massenspeicher (FDD, HDD, CD-ROM ...)
  - USB-Schnittstellen, Firewire-Schnittstellen (IEEE-1394)
  - Integrierte Grafik (VGA, DVI, HDMI), Audio, WLAN
- Die Hauptplatine ist der eigentliche Rechner

PC-Aufbau

- Grafikkarten: Steckkarten oder integriert (On-Board-Graphics)
  - Auflösung normal bis 1600 x 1200, evtl. mehr
- Monitor: früher CRT, heute TFT (Thin-Film Transistor, Matrix)
- Speichermedien (permanent):
  - magnetisch: Diskette, Festplatte, standardmäßig bis 1000 GB
  - optisch: CD, DVD, Blue-Ray, HD-DVD (bis 25 GB/Schicht)
  - Solid-State (EEPROM): USB-Stick, Speicherkarten
- Tastatur, Maus usw.: 8051-Einplatinenrechner
- Erweiterungskarten: Netzwerk, Firewire, Modem/Fax, Messkarten
- Innen-Schnittstellen: IDE, SATA, SCSI
  - Für magnetische/optische Speichermedien; SATA2 bis 3 Gbit/s
- Aussen-Schnittstellen: seriell (RS232C, PS/2, USB, Firewire), parallel, eSATA

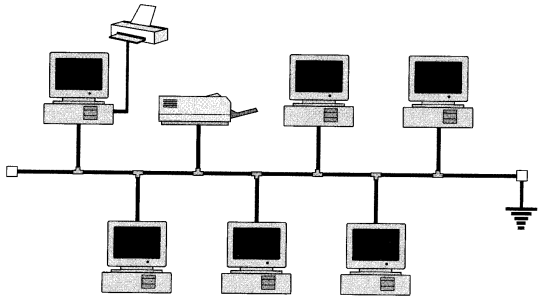
PC-Aufbau

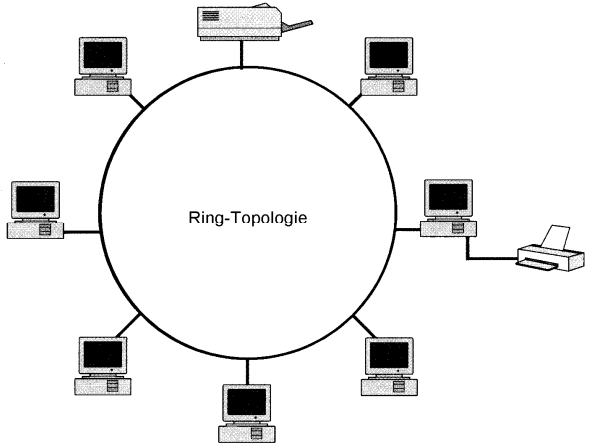
- Netzwerkkarten
  - BNC (10 Mbit/s), TwistedPair (10/100/1000 Mbit/s), Glasfaser
  - WLAN (wireless LAN): 11/54/108 Mbit/s (54: IEEE 802.11g)

- Datensicherung
  - Bandgeräte
  - CD/DVD-R/RW bis 4/8 GB pro DVD
- Preiswert, einfach, schnell, bequem: externe USB/eSATA-Festplatten
  - etwa 60 EUR für 1000 GB in 3.5“, für 500 GB in 2.5“
- Im Uni-Hannover-Bereich
  - Archivierungsserver (UQFS), Zugang übers Netz, Technik: Bandroboter

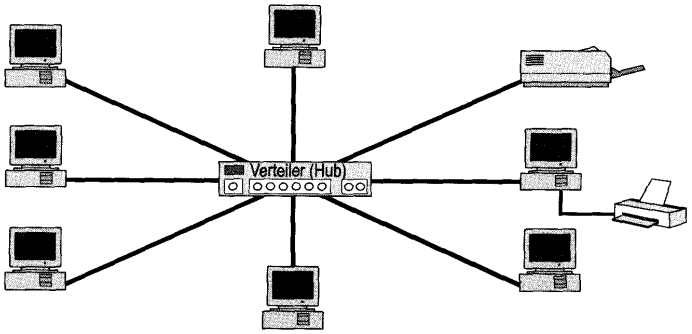
Netzwerke I – Physikalische Topologie

- Ein Netzwerk ermöglicht einen universellen Datenaustausch unter allen daran beteiligten Computern
  - Bustopologie (Ethernet, ThickWire oder ThinWire; AppleTalk)
  - Ringtopologie (Token Ring)
  - Sterntopologie (Ethernet, Twisted-Pair mit Hubs)
  - Baumtopologie - Mischung aus Bus- und Sterntopologie (fast alle)

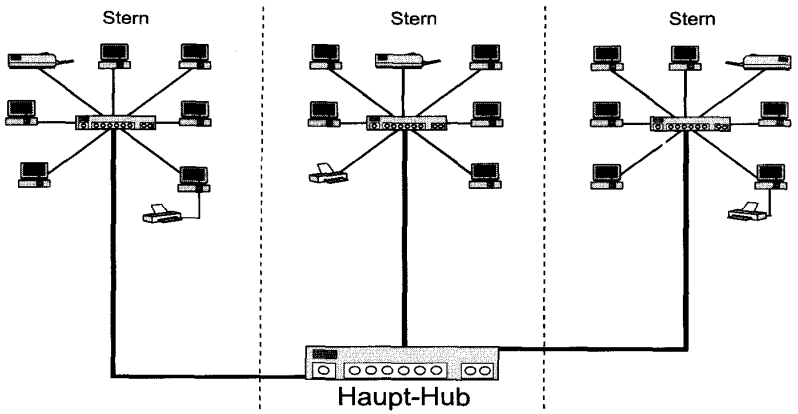




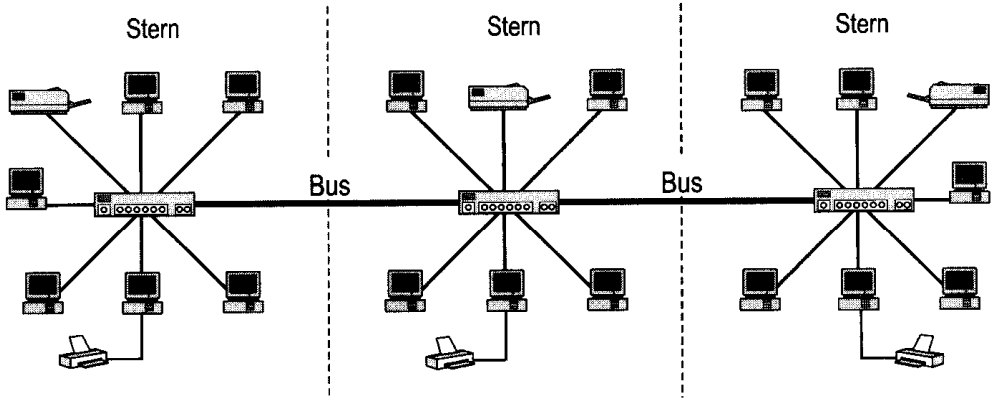
Netzwerke II – Physikalische Topologie



Stern-Topologie



Stern-Stern-Netz



Stern-Bus-Netz

Netzwerke III

- Verwendet wird fast ausschließlich: Ethernet -Verfahren
  - 10 Mbit/s (Ethernet)
  - 100 Mbit/s (Fast-Ethernet): Benutzernetz Uni Hannover
  - 1000 Mbit/s (Gigabit-Ethernet): Backbones

- Realisierung
  - Twisted-Pair (paarweise verdrillte Kabel)
  - 100Base-T: aktuelles Standard in LANs
    - Sterntopologie (physisch), Bus (logisch), 10/100 Mbit/s
    - Max. 1024 Knoten pro Segment, max. 100 (150) m vom Hub entfernt
  - LWL (100/1000Base-F): für längere Strecken bzw. Backbones eingesetzt
    - Sterntopologie (physisch), Bus (logisch), bis 1000 Mbit/s
    - Max. 1024 Knoten pro Segment, max. 2000 m vom Hub entfernt

Netzwerke IV

- Anwendungsgebiete
  - Datei/Drucker-Sharing
  - Verwaltungsaufgaben
  - Informationszugriff (Internet, Datenbanken)

- Übertragungsprotokolle IP, TCP (TCP/IP)
  - Sinn: fehlerfreie Informationsübertragung sicherstellen
  - eindeutige Adressierung (IPv4): 130.75.115.40; 2^{32} Adressen (4,29 Mld.)
  - DNS: Domain Name Service (travel.tci.uni-hannover.de)
  - Private Bereiche: 10.x.x.x, 172.16.x.x-172.31.x.x, 192.168.x.x
  - IPv6: 6-stellige IP-Adresse, 2^{128} Adressen

WLAN-Netzwerke

- Drahtlose Kommunikation im 2.4 bzw. 5.4 GHz-Band
- IEEE 802.11a, h (5.4 GHz), 802.11b, g (2.4 GHz)
- 802.11g die aktuelle Variante, 13 Kanäle
- Übertragungsrate (g) 6-108Mbit/s brutto, 2-32 Mbit/s netto
- Datendurchsatz von der Signalstärke abhängig
- Reichweite max. 100 m im Freien
- Störung durch BlueTooth, schnurlose Telefone, Mikrowellenherde
- Datensicherheit: WPA (WiFi Protected Access), WEP (Wired Equivalent Privacy), WPA, WPA2, MAC-Adresse
- Peer-To-Peer oder Access-Point-Topologie
- Uni-Hannover: Netzzugang auf verschiedenen Standorten (Liste auf dem RRZN-Server)

Rechner-Software

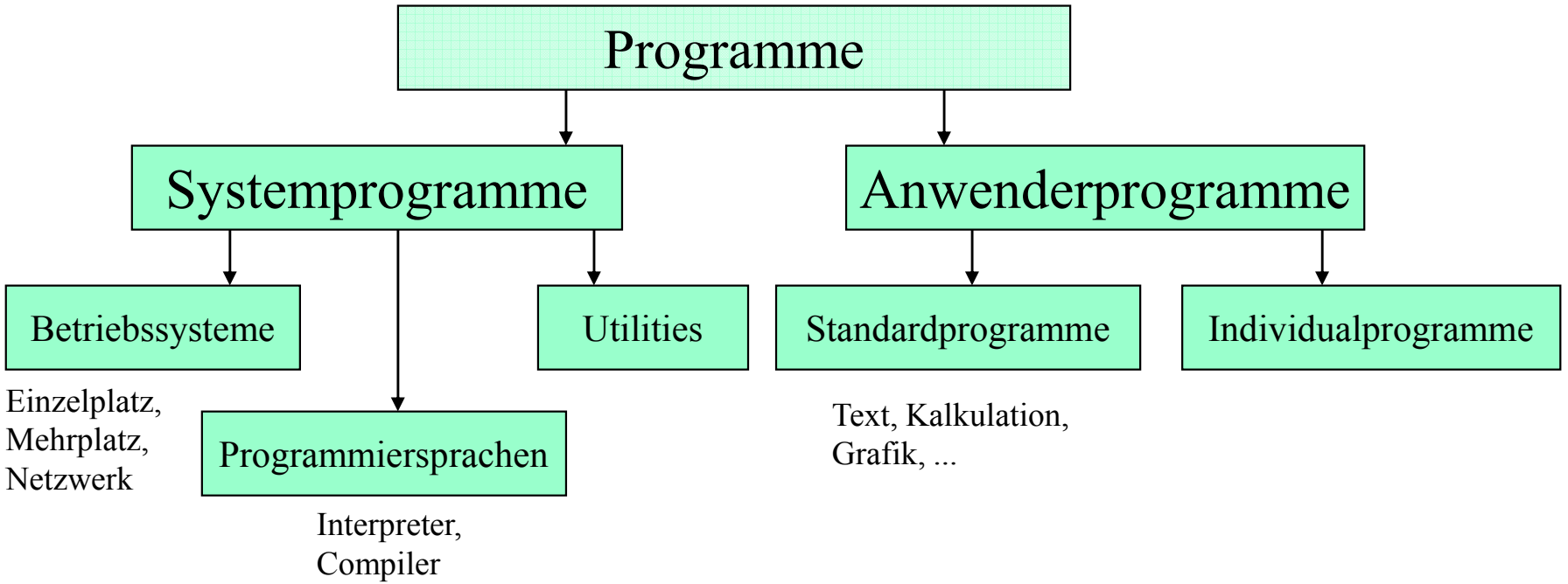
- BIOS: zwischen Hardware und Betriebssystem

- Systemprogramme
  - Betriebssysteme (Unix, Linux, Windows, DOS)
  - Utilities (Winzip, Total Commander)
  - Programmiersprachen (C, Java, …)

- Anwenderprogramme
  - Fertig geliefert: Office, Maple, Photoshop, Acrobat …
  - Eigenbau: Programmierung

Rechner-Betriebssysteme

- Betriebssystem: ein Programm, das die Verbindung zwischen dem Benutzer, der Anwendersoftware und der Hardware herstellt
- Benutzer – Anwendung – Betriebssystem – Rechnerhardware



Betriebssysteme verschiedener Rechner

- Mikroprozessor-Rechner, Einplatinen-Rechner
  - Basic – verschiedene Varianten

- PC
  - MS-DOS und andere DOS-Varianten (DR-DOS usw.)
  - Windows 3.x, 95, 98, ME, NT 3.51, NT 4.0, XP, Vista
  - Linux, Unix
  - Industriebetriebssysteme mit Multitasking

- Workstations (Alpha, HP, Sun, VAX)
  - Unix in verschiedenen Varianten (Sun: Solaris), VMS (Alpha, VAX)
  - Prozessdatenerfassung in Technikum-TCI: VAX-Maschinen mit VMS

Programmiersprachen

- Zweck einer Programmiersprache
  - Erleichterung der Kodierung der Aufgabenlogik
- Klassifizierung – verschiedene Kriterien
  - Komplexität der Grundanweisungen
    - Low-Level : Code, Assembler
    - Höhere Programmiersprachen: Basic, Fortran, C, Pascal, Delphi
    - Strukturierte Programmiersprachen: Ada, C++, MATLAB
  - Systemnähe (Hardware-Nähe)
    - Systemsprachen: Assembler, C
    - Anwendungssprachen: Basic, C, C++, Fortran, Pascal, Delphi, Java, MATLAB
  - Objektorientierung
    - Nichtobjektorientierte
      - Assembler, C, Basic, Fortran, Pascal, Ada
    - Objektorientierte
      - C++, Java, MATLAB

Klassifizierung der Programmiersprachen nach Generationen

- Erste Generation: Maschinensprachen
  - Binärzahlen – Prozessorbefehle
  - Beispiel: C3 00 01 (JMP 100h, Instruktion von Intel 8080A)
- Zweite Generation: Assembler-Sprachen
  - An bestimmte Prozessoren mit ihrer Logik gebunden
  - Hardwarenahe Befehle
  - Operationskürzel, Segmentierung, höhere Befehle: Schleifen usw.
  - Beispiel: MOV A,M (Inhalt einer Speicherposition kommt in den Register A)
- Dritte Generation: Problemorientierte (prozedurale) Sprachen
  - Auch „höhere“ Programmiersprachen genannt
  - Programmierer beschreibt die genaue Prozedur, wie man zum Ergebnis kommt
  - Unabhängig vom Prozessor, Betriebssystem
    - Compiler für konkreten Prozessor, Betriebssystem notwendig
  - Sprachen für bestimmte Problembereiche angepasst
  - Cobol, Fortran, Pascal, PL/1, Basic, Ada, C, C++, Java, Delphi u.v.a.
  - Beispiel: x21 = b25-b25*ab; ab=ab+1 (Fortran); x21 = b25 - *ab++ (C)

Klassifizierung der Programmiersprachen nach Generationen

- Vierte Generation: Deklarative Sprachen
  - Für spezielle Anwendungsgebiete
    - Tabellenkalkulationen (Excel)
    - Datenbanken (SQL)
    - Algebraische Manipulationen (Maple)
    - Prozessberechnung und -simulation (Matlab)
  - Programmierer beschreibt die gewünschte Aktion, aber nicht den Weg dazu
  - Beispiele: SQL, Access, Maple, Matlab
- „Fünfte Generation“: Künstliche Intelligenz
  - Logische Zusammenhänge werden ausgewertet
  - Lisp, Prolog, G2

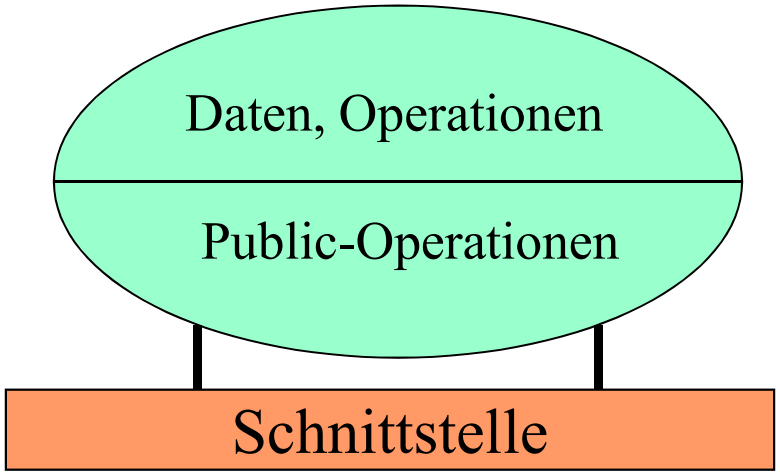
Klassifizierung nach Sprachtyp

- Prozedurale Programmiersprachen
  - Fortran, Cobol, Basic, C, C++, Pascal, Delphi, Java, Assembler, MATLAB, Excel, Maple, SQL – alle „Normalgebrauch“-Sprachen
    - Typische Ablaufstruktur
      - StartProgramm
        - » Call Prozedur-1
        - » Anweisung-1
        - » Anweisung-2
        - » Call Prozedur-2
      - FinishProgramm
    - Zerlegung der Aufgaben auf Schritte, die in Unterprogrammen (Prozeduren) untergebracht werden und repetitiv benutzt werden können
    - Modulare Programmierung: Anwendung fertiger Module

- Logische Programmiersprachen
  - Prolog: Fakten und Regeln werden eingegeben, Abfrage liefert Ergebnis

Klassifizierung nach Sprachtyp

- Objektorientierte und nicht-objektorientierte Sprachen
  - Objektorientierung: Weiterentwicklung der modularen Programmierung, Abkapselung der Prozeduren -> Objekte
    - Es werden Objekte definiert, die Folgendes enthalten (können)
      - Datenstrukturen
      - Funktionen
      - Andere Objekte
  - C++ (C-Syntax), Java (portabel), Smalltalk (interpretiert), Matlab

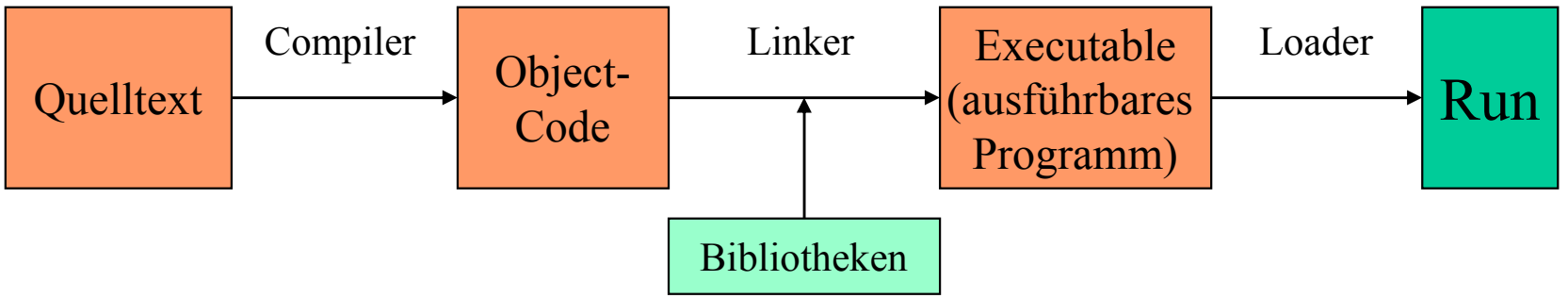


Programmierung

- Werkzeuge der Softwareentwicklung
- Syntaktische Elemente einer Programmiersprache
  - als Beispiel: Programmiersprache C (Grundlage für C++, C#, Java)
  - Deskriptoren
  - reservierte Wörter
  - Kommentare
  - Datentypen, Variablen, Konstanten
  - Operatoren
  - Ausdrücke, Anweisungen
  - Kontrollstrukturen
  - Unterprogramme: Funktionen, Prozeduren
  - Eingabe/Ausgabe (I/O-Funktionen)

Werkzeuge der Software-Entwicklung

- Editor
- Compiler (Übersetzer)
- Linker, Loader
  - Compiler + Linker in einem zur Laufzeit: Interpreter (Basic)
- Standardbibliotheken
- Debugger
- Entwicklungsumgebung (IDE, Integrated Development Environment)
    - Editor, Compiler, Linker+Loader, Bibliotheken, Debugger in einem



Programm: Strukturbeispiel, Programmiersprache C

```
#include <stdio.h>                /* predefined I/O functions  */
double c_to_f (double);           /* function declaration */
int main (int argc, char *input_arguments[]); /* main program is a function in C */
{ int i,j,k;                                  /* variable declaration/definition */
  double c, dc, f, start_c, ca[100], fa[100];   /* variable declaration/definition */
  for (i=0, start_c=0, dc=20; i < 5; i++)       /* loop 5 times, increment 20 deg C   */
    { c = start_c + (double)i * dc;           / * c = deg C       */
      f = c_to_f(c);                          /* convert to deg F   */
      fa[i] = f;                              /* put the value into array   */
    }
   i = 0; do {printf("%8.2f\n",fa[i]); i++;} while (i < 5);  /* Print results */
} /* End of function main()  */
double c_to_f (double celsius)            /* definition of the c_to_f function */
  { const double k9=9, k5=5, k32=32;      /* define conversion constants */
    return (celsius * k9/k5) + k32;       /* return celsius*(double)9/5+32 possible */
  }
```

Syntaktische Elemente einer Programmiersprache

- Wird auf dem Beispiel der Programmiersprache C dargestellt

- Deskriptoren (Bezeichner): eindeutige Identifizierung von Objekten
  - Namen für Konstanten, Variablen, Typen, Prozeduren
- Reservierte Wörter: vordefinierte Bedeutung, können nicht als Objektnamen verwendet werden
  - double, int, char, break, case, do, if, else, for, return, continue, while usw.
- Kommentare
  - /* Kommentartext */
- Variablen
  - der Wert einer Variable kann währen des Programmablaufs geändert werden
    - Globale, lokale, statische, externe - Verhalten während des Ablaufs
    - Deklaration, Definition, Wertzuweisung: int a, b=5, c=10; a = b*c;
  - Typenqualifizierer const: der Variablenwert kann nicht geändert werden
    - const double ac = 0.15;

Variablen

- Basistypen
  - Numerische, Zeichen, Leer
    - int i = 3; /* Integer - ganzzählig */
    - double x = 12.4; /* Float - Fließkomma */
    - char c='p', ca[]="abcde"; /* Character - Zeichen */
    - void fl(); /* Leer - Funktionen */

- Zusammengesetzte Datentypen (über die Basistypen definiert)
  - Zeiger: zu jedem Basistyp existiert ein Zeigertyp
  - Wert eines Zeigers = Adresse eines Objekts
  - int five = 5, j; int *p5;
  - p5 = &five /* Adresse der Variable five */
  - j = *p5 /* Dereferenzierung, j = five */

Variablen

- Zusammengesetzte Typen - Fortsetzung
  - Felder (arrays): Indexierung ab 0
  - int vector[10]; /* vector[0] ... vector[9] */
  - vector[9] = 13;
  - double mat[10][20]; mat[i][j]=32.1;

  - Strukturen: struct complex {double re; double im;} xc;
  - xc.re = 2.5; xc.im = 3.1;

  - Bitfelder: Bitfelder in einer Variable bekommen Namen
    - Prozessornahe Programmierung

  - Union (Vereinigung): Komponenten einer Struktur überlagern sich
    - z.B, eineVariable wird gleichzeitig als Character und als Integer verwendet

Ausdrücke

- Ausdrück: eine Folge von Operatoren und Operanden, die
  - einen Wert berechnen: b * (c + d); ((a > 100) && (b < 500)) || (c > 1)
  - einen Objekt bezeichnen: a; b=ft(x);
  - Typ eines Objekts festlegen: x = (double)a/b;
  - Ausdruck in C hat immer einen Typ und einen Wert
- Operanden (in etwa = Variablen): sind selbst Ausdrücke
  - a *= (b+c); /* eq. a = a * (b+c) */
- Operatoren: definieren die Operation zwischen den Operanden
  - Arithmetische (+, -, *, /, %, ++, --, ^)
  - Vergleichsoperatoren (==, !=, <, >, <=, >=)
  - Logische Operatoren (&&, ||, !, ^ bzw. and, or, not, xor)
  - Bitweise Operatoren (~, |, &, ^, >>, <<)
  - Zuweisungsoperatoren (=, +=, -=, *=, /= usw.)
  - unäre, binäre, ternäre; Priorität, Reihenfolge, Klammern

Ausdrücke-Anweisungen (Statements)

- Anweisungen
  - Anweisung: die kleinste ausführbare Einheit

- Einfache Anweisungen:
  - leere Anweisung ";" (syntaktische Verwendung): for (i=0;i<9;i++);
  - Zuweisunge, Funktionsaufrufe: i = 1; a = max(b,c); a = b * (c + d);
  - Anweisungsblöcke: {deklarationen; anweisungen;}

- Schleifen, Sprunganweisungen, Auswahlanweisungen
  - Kontrollstrukturen - steuern den Programmfluss
  - Schleifen: Wiederholung bestimmter Anweisungsblöcke
  - Sprunganweisungen: Springen ohne weitere Bedingung
  - Auswahlanweisungen: Zwischenergebnisse bestimmen die Fortsetzung

Kontrollstrukturen: Schleifen

- Kontrollstrukturen
  - Schleifen (Wiederholung): while, do-while, for
  - Sprunganweisungen: break, continue, return, goto
  - Auswahlanweisungen: if, if-else, else-if, switch-case

- Schleifen
  - while (Ausdruck) Anweisung;
  - do {Anweisung;} while (Ausdruck): wird mind. 1x ausgeführt
  - for-Schleife: übersichtliche while-Schleife

  - i=0,j=0; while (i<5) {i++; j+=5;}
  - i=0,j=0; do {i++; j+=5;} while (i<5);
  - for (i=0,j=0; i<5;i++) j+=5;

Kontrollstrukturen: Sprunganweisungen

- Sprunganweisungen
  - break: die innerste Schleife in while/do/for/switch wird verlassen
  - continue: die nächste Schleifeniteration wird fortgesetzt
  - return: die Funktion wird verlassen, mir Rückkehr zu der aufrufenden Stelle
  - goto: Sprung zu einer Marke (label) im Programm

  - i=0; while (i<5)
    {
    i++; if (i = = 3) break;
    }

Kontrollstrukturen: Auswahlanweisungen

- if, if-else, else-if, switch: konditionale Anweisungen
- die Ausführung einer Anweisung wird von dem Wert eines Ausdrucks abhängig gemacht

  - if (i = = 5) j = 3; /* on i != 5, j undefined */
  - if (i = = 5) j = 3; else j = 1; /* j has always a value */
  - if (i = = 5) j = 3;
    else if (i = = 2) j = 4;
    else j = 0;
  - switch (i) /* same as before */
    { case 5: j = 3; break;
    case 2: j = 4; break;
    default: j = 0; break;
    }

Funktionen/Prozeduren: Unterprogrammtechnik

- Funktionen/Prozeduren: Unterprogrammtechnik
  - für sich wiederholende Aufgaben
  - Zerlegung des Programms in selbstständige Einheiten (Übersichtlichkeit, Wartbarkeit)
  - Funktionsargumente (Parameter)
  - Rückgabewert: ein oder mehrere; die letzte Funktionsanweisung
  - Parameterübergabe als Wert (call by value), als Zeiger (call by reference)
  - double dbmax (double a, double b) /* Maximum-Funktion */
    { if (a > b) return a;
      else return b;
    }
  - int main () /* Funktionsaufruf */
    { double x=5, y=2, z;
      z = dbmax(x,y);
    }

I/O-Funktionen

- I/O-Funktionen (Eingabe/Ausgabe)
  - Werteeingabe von der Tastatur, aus Dateien
  - Werteausgabe auf den Bildschirm, Drucker, in eine Datei
  - spracheabhängige Funktionen
  - scanf(format, &var), fscanf(format,&var): Eingabe
  - printf(format,var), fprintf(format,var): Ausgabe
  - Formatierung: %d, %f, %s für Integer, Float, String

```
int i;
scanf("%d",&i);                 /* Enter integer  */
printf("Integer: %d \n",i);     /* Output integer  */
```

  - dptr = fopen("abc.dat","w"); /* Open file abc.dat for write */

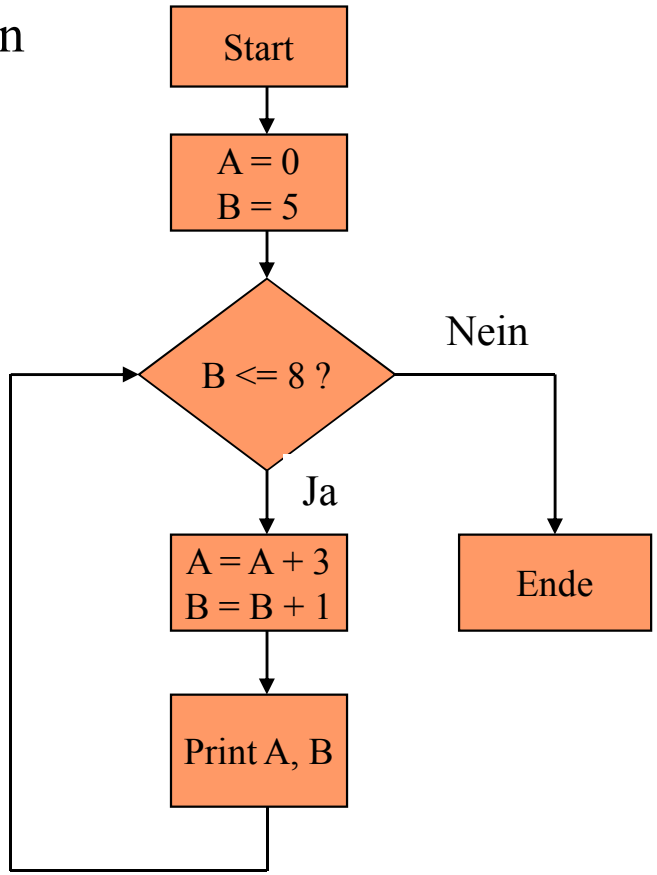
```
fprintf (dptr,"%d \n", i);      /* Print one integer  */
fclose (dptr);                  /* Close the file       */
```

Literatur

- Skripte beim RRZN: Angebote/Handbücher
  - http://www.rrzn.uni-hannover.de/buecher.html
- Skript - Hardware: PC-Technik Grundlagen
- Skript: Grundlagen der Programmierung
- Skript: Netzwerke

Programmbeispiel 1 - Programmerstellung I

- Programmablaufplan



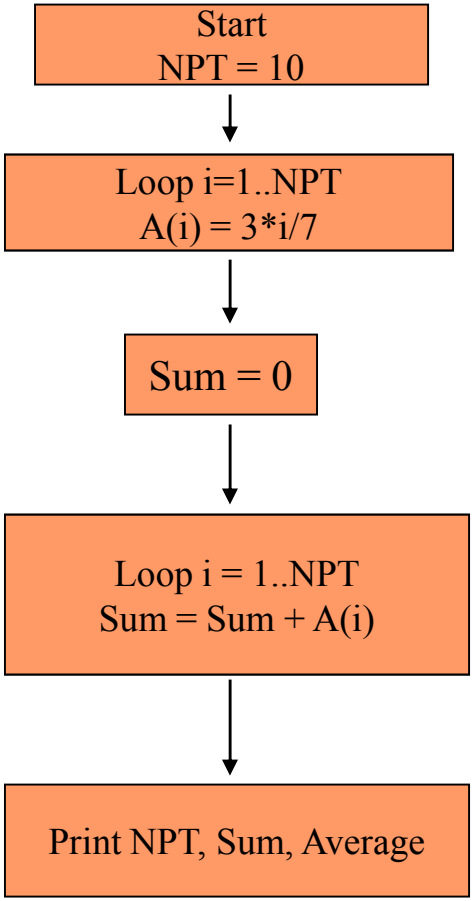
Programmbeispiel 1 - Programmerstellung II

- Pseudocode
  - Beginne Prozedur
    - Initialisiere A=0, B=5
    - Wiederhole
      - {A = A + 3; B = B + 1; Print A, B } bis B > 8
  - Beende Prozedur
- Programmerstellung (C)
    - int main()
      {int a=0, b=5;

      while (b <= 8)
       {a += 3; b++; printf (“a, b: %3d, %3d\n“);}
       }

Programmbeispiel 2

- Programmablaufplan



Übung – Grundlagen der Programmierung

- 2 Aufgaben - geleitet mit Erläuterung
  - einfache Schleife mit Ausgabe (aus Getting Started ...)
  - Konversion Celsius-zum-Fahrenheit
    - Schleife
    - Ausgabe
    - Unterprogramm

- Start/Programme/Open Watcom C-C++ Tools Help/IDE Help

  Getting started with a short tutorial

- Start/Programme/Open Watcom C-C++/IDE
  - Öffnen der Programmierumgebung: Text-Editor, Kompilierung, Linken

Übung – Grundlagen der Programmierung

```
#include <stdio.h>

    void main()
    {
       int i;
       for (i=0; i<10; i++)
       {
          printf( "Value is %d\n", ii );/* Deliberate error to show IDE reaction */
       }
       printf("Press Enter...\n");
       getchar();
    }
```

Übung – Grundlagen der Programmierung

```
#include <stdio.h>  /* predefined I/O functions  */

double c_to_f (double);         /* function declaration */

void main ();                                   /* main program is a function in C */
{ int i,j,k;                            /* variable declaration/definition */
  double c, dc, f, start_c, ca[100], fa[100];   /* variable declaration/definition */

  for (i=0, start_c=0, dc=20; i < 5; i++)       /* loop 5 times, increment 20 deg C   */
    { c = start_c + (double)i * dc;           / * c = deg C       */
      f = c_to_f(c);                          /* convert to deg F   */
      fa[i] = f;                              /* put the value into array   */
    }

   i = 0; do {printf("%8.2f\n",fa[i]); i++;} while (i < 5);  /* Print results */
   printf("Press Enter...\n");  getchar();   /* Wait for keypress ...  */
} /* End of function main()  */
```

Übung – Grundlagen der Programmierung

Fortsetzung in der gleichen Quelldatei:

```
double c_to_f (double celsius)          /* definition of the c_to_f function */

{
   const double k9=9, k5=5, k32=32;     /* define conversion constants */
   return (celsius * k9/k5) + k32;      /* return celsius*(double)9/5+32 possible */
 }
```