Hochschule **Rosenheim**
University of Applied Sciences

Prozedurale Programmierung Einführung

Technische Hochschule Rosenheim

WS 2018/19

Prof. Dr. F.J. Schmitt

Einführung

- Zielsetzung:
  - Systematisches Problemlösen
  - Programmiermethodik
  - Typsystem, Kontrollstrukturen, Funktionen
  - Schrittweises Erlernen der Programmierung am Beispiel von C

- Methode:
  - Vorlesung und Praktikum
  - Kennenlernen von Code-Beispielen

  - **Selbststudium**: Selbständiges Lösen von Aufgaben – Erstellung von C-Programmen

Organisatorisches (1)

- Vorlesungen
  - 4 Wochenstunden
  - Montag und Mittwoch
- Übungen
  - 2 Wochenstunden
  - Mittwoch und Donnerstag, Raum A0.03
  - Aufteilung in **vier** Gruppen
  - Visual Studio, Windows 8.1, VMWare
- Download der Vorlesungs- und Übungsunterlagen
  - über INF Community

Organisatorisches (2)

- Hinweise zur Leistungserbringung:
  - Klausur zu Semesterende
  - 90 Minuten
  - Unterlagen: keine
  - Anmeldung über Online-Service-Center notwendig (November)

- Ansprechpartner:
  - Email: fj.schmitt@fh-rosenheim.de
  - Büro: B 1.19

Literatur

- Dausmann, M., Bröckl, U., Goll, J. (2010):
  C als erste Programmiersprache,
  Vieweg+Teubner, 7. Auflage
  (E-Book, Bibliothek, WebOpac)

- Klima, R., Selberherr, S. (2010):
  Programmieren in C,
  Springer, 3. Auflage
  (E-Book, Bibliothek, WebOpac)

- Kernighan, B.W., Ritchie, D. M. (1988):
  The C Programming Language,
  Prentice Hall International, 2. Auflage

- Wolf, J. (2009):
  C von A bis Z,
  Galileo Computing, 3. Auflage,
  http://openbook.galileocomputing.de/c_von_a_bis_z/

Inhaltsverzeichnis (1)

Inhaltsverzeichnis (2)

Überblick Kapitel 1

- Klärung wichtiger Begriffe
  - Programm / Programmieren / Programmiersprachen
  - Typen und Typsystem
  - Algorithmus

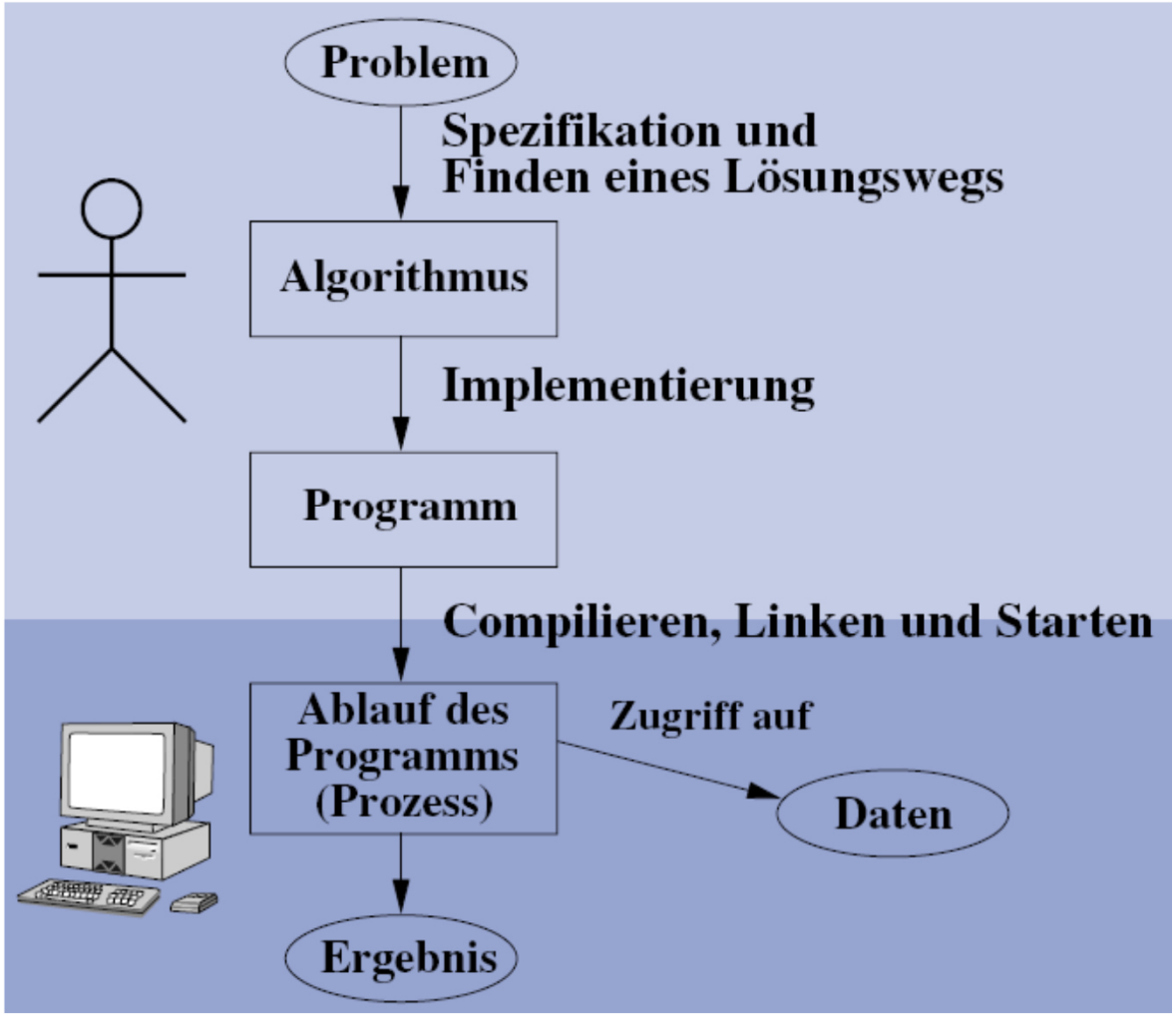
- Prozess der Programmerstellung und –ausführung
  - Systematische Vorgehensweise
  - Werkzeuge (Compiler, Linker, Lader, Debugger, integrierte Entwicklungsumgebung)

Was ist die Ausgangssituation?

Problem, das zu lösen ist

Problemlösung soll durch Computer ausgeführt werden

Wie ist die Aufgabenverteilung?



Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Schritt 1: Problembeschreibung (1)

- Spezifikation der Aufgabenstellung, d.h. das zu lösende Problem ist genau zu beschreiben
  - vollständig: alle relevanten Informationen sind berücksichtigt
  - detailliert: alle Hilfsmittel und Grundaktionen sind aufgelistet, die zur Lösung zugelassen sind
  - eindeutig: Festlegung von klaren Kriterien, wann eine Lösung akzeptabel ist

Schritt 1: Problembeschreibung (2)

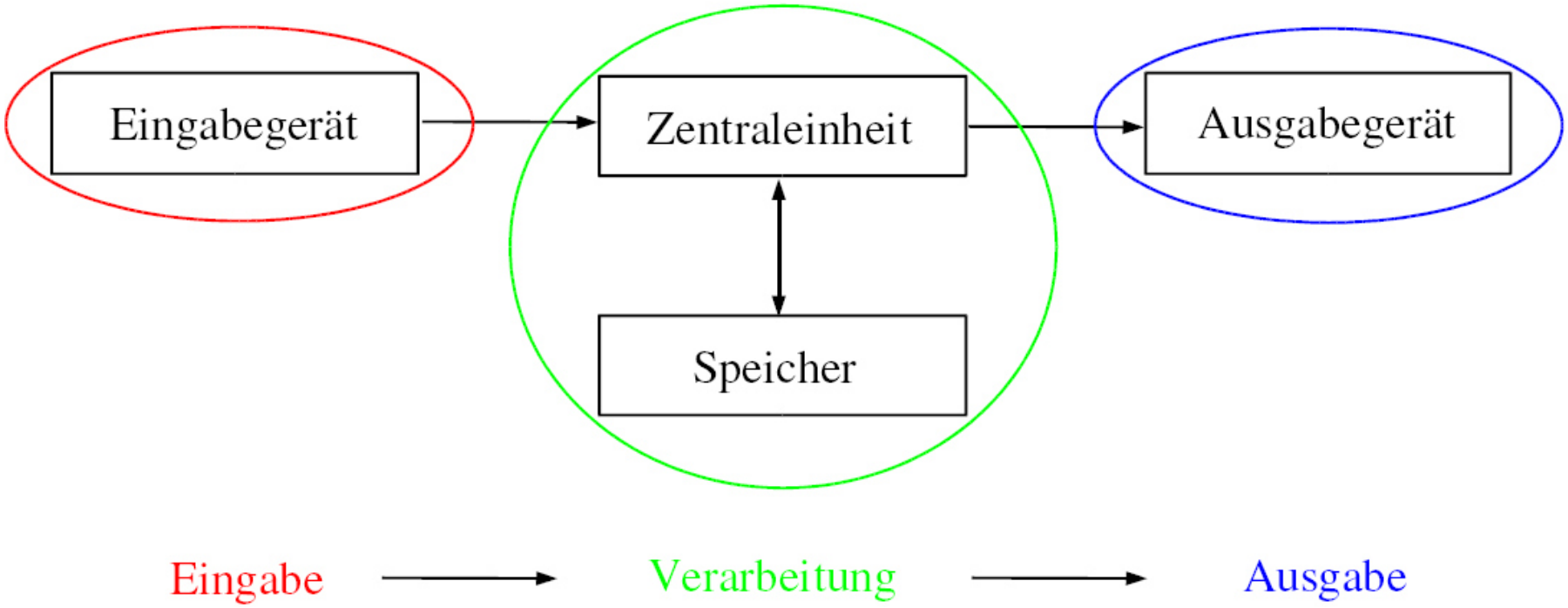
- Beispiel 1:

Zu zwei Zahlen X und Y ist der größte gemeinsame Teiler ggT(X,Y) zu berechnen.

Sehr ungenaue Spezifikation!

Problemlösung soll durch Computer ausgeführt werden:

EVA-Prinzip bei Computern



Schritt 1: Problembeschreibung (3)

- Bessere Spezifikation:

Zu zwei positiven natürlichen Dezimalzahlen X und Y,
die über Tastatur einzulesen und
jeweils in einem Speicher von 4 Bytes zu speichern sind,
ist der größte gemeinsame Teiler ggT(X,Y) zu berechnen.
Das Ergebnis der Berechnung ist auf dem Bildschirm auszugeben,
wobei die Inhalte von X und Y nach der Berechnung weiterhin ihre alten Werte besitzen müssen.

Schritt 2: Finden eines Lösungswegs

- Ziel: Entwerfen eines Lösungswegs, bei dem jeder auszuführende Schritt genau beschrieben ist

- Erfolgt mittels eines so genannten Algorithmus
  - Vorschrift zur schrittweisen Lösung eines Problems
  - Abgeleitet vom Namen eines berühmten persischen Buchautors (Al-Khwarizmi, 825 n.Chr.)

Beispiel: Rezept

Rinderbeinscheiben in Rotweinsauce

Eingabe

Zutaten

2	Karotten
400 g	Knollensellerie
1	Kohlrabi
4	Rinderbeinscheiben (à ca. 5 cm dick)
3 EL	Öl
¼ l	Rotwein (trocken, z. B. Spätburgunder)
¼ l	Rinderbrühe
1–2 TL	Speisestärke
	Salz | Pfeffer

Verarbeitung

1 Das Wurzelgemüse putzen und waschen bzw. schälen und klein schneiden. Die Beinscheiben waschen und trocken tupfen. Das Öl in einem Bräter erhitzen und das Fleisch darin auf beiden Seiten anbraten. Das Gemüse hinzufügen und kurz mitbraten. Mit Wein und Brühe aufgießen und 30 Min. köcheln lassen.

2 Das Fleisch aus dem Bräter nehmen. Die Sauce durch ein Sieb in einen Topf gießen. Die Stärke mit wenig kaltem Wasser glatt rühren und die Sauce damit binden. Die Sauce mit Salz und Pfeffer und nach Belieben noch mit etwas Rotwein abschmecken. Die Rinderbeinscheiben mit der Sauce auf vier Tellern anrichten. Dazu werden im Gasthof »Zur Einkehr« Gemüsetaler und ein grüner Salat serviert. Aber auch Blaukraut und Kartoffelklöße sind eine passende Beilage.

Ausgabe

Rezept aus:
C. Brauer, M. Müller (Hrsg). Gscheitgut, Franken isst besser. Michael Müller Verlag, Erlangen. 2. Auflage 2012

Definition Algorithmus (nach Knuth)

Hochschule **Rosenheim**
University of Applied Sciences

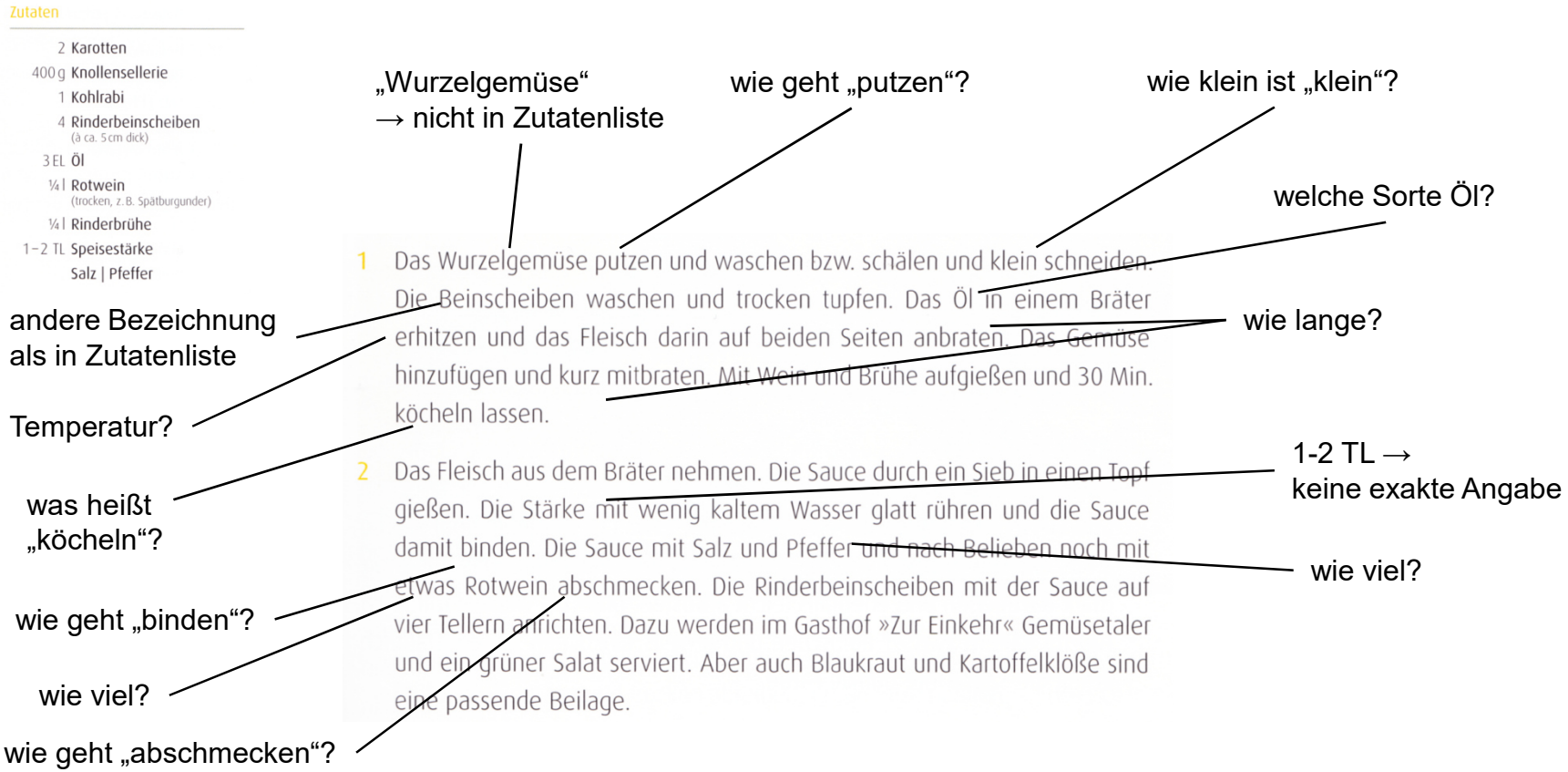
- Endlichkeit
  - ein Algorithmus muss immer nach einer endlichen Anzahl Schritten terminieren
- Bestimmtheit
  - jeder Schritt eines Algorithmus ist in jedem Fall eindeutig definiert
- Eingabe
  - ein Algorithmus hat Eingabeparameter (statisch oder dynamisch)
- Ausgabe
  - ein Algorithmus hat mindestens einen Ausgabewert, der sich aus den Eingabeparametern ableitet
- Effektivität
  - Anweisungen müssen grundlegend genug sein, so dass sie prinzipiell
    - exakt und
    - in endlicher Zeit ausgeführt werden können

Algorithmus
Anmerkungen

- es gibt Beispiele für nicht-terminierende Algorithmen (Regelschleifen in eingebetteten Systemen, Betriebssysteme)
  - streng genommen kein Algorithmus
  - Knuth schlägt hierfür den Begriff „Computational Method“ vor.
- effektiv ≠ effizient
- Endlichkeit ist für praktische Zwecke ein schwaches Kriterium; ein Algorithmus soll auch effizient sein
- Das Problem, ob ein Algorithmus terminiert, ist unentscheidbar → es gibt keinen Algorithmus, der für einen beliebigen Algorithmus entscheiden kann, ob er terminiert

Ist das Rezept wirklich ein Algorithmus?

Hochschule **Rosenheim**
University of Applied Sciences



Zutaten

- 2 Karotten
- 400 g Knollensellerie
- 1 Kohlrabi
- 4 Rinderbeinscheiben (à ca. 5 cm dick)
- 3 EL Öl
- ¼ l Rotwein (trocken, z. B. Spätburgunder)
- ¼ l Rinderbrühe
- 1–2 TL Speisestärke
- Salz | Pfeffer

1 Das Wurzelgemüse putzen und waschen bzw. schälen und klein schneiden. Die Beinscheiben waschen und trocken tupfen. Das Öl in einem Bräter erhitzen und das Fleisch darin auf beiden Seiten anbraten. Das Gemüse hinzufügen und kurz mitbraten. Mit Wein und Brühe aufgießen und 30 Min. köcheln lassen.

2 Das Fleisch aus dem Bräter nehmen. Die Sauce durch ein Sieb in einen Topf gießen. Die Stärke mit wenig kaltem Wasser glatt rühren und die Sauce damit binden. Die Sauce mit Salz und Pfeffer und nach Belieben noch mit etwas Rotwein abschmecken. Die Rinderbeinscheiben mit der Sauce auf vier Tellern anrichten. Dazu werden im Gasthof »Zur Einkehr« Gemüsetaler und ein grüner Salat serviert. Aber auch Blaukraut und Kartoffelklöße sind eine passende Beilage.

Fazit:

- damit könnte ein Computer nichts anfangen
- viele Menschen übrigens auch nicht – obwohl das ein gut ausgearbeitetes Rezept ist
- man muss dem Computer **alles** haarklein beschreiben

Formulierung eines Algorithmus

- Informelle Beschreibung mittels Text

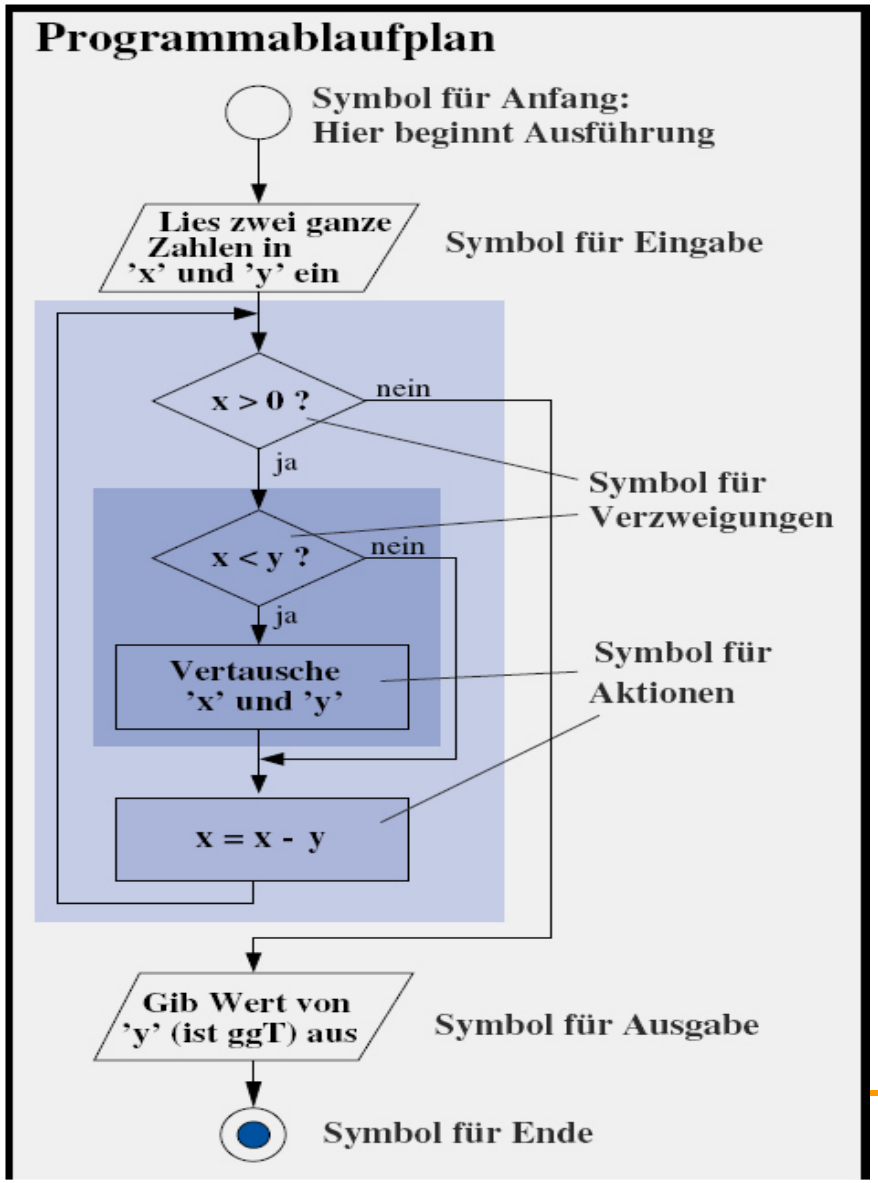
Algorithmus zum Ermitteln der größten Zahl aus einer Menge von Zahlen:

1: Lies die erste Zahl!
2: Speichere diese Zahl in der Variablen 'max'!
3: Solange nicht alle Zahlen gelesen, wiederhole Schritte 3.1 bis 3.2!
   3.1: Lies die nächste Zahl!
   3.2: Wenn diese Zahl größer als Zahl in der Variablen 'max' ist, speichere diese Zahl als neuen Wert in Variablen 'max'!
4: Gib den Wert der Variablen 'max' aus!

Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Formulierung eines Algorithmus

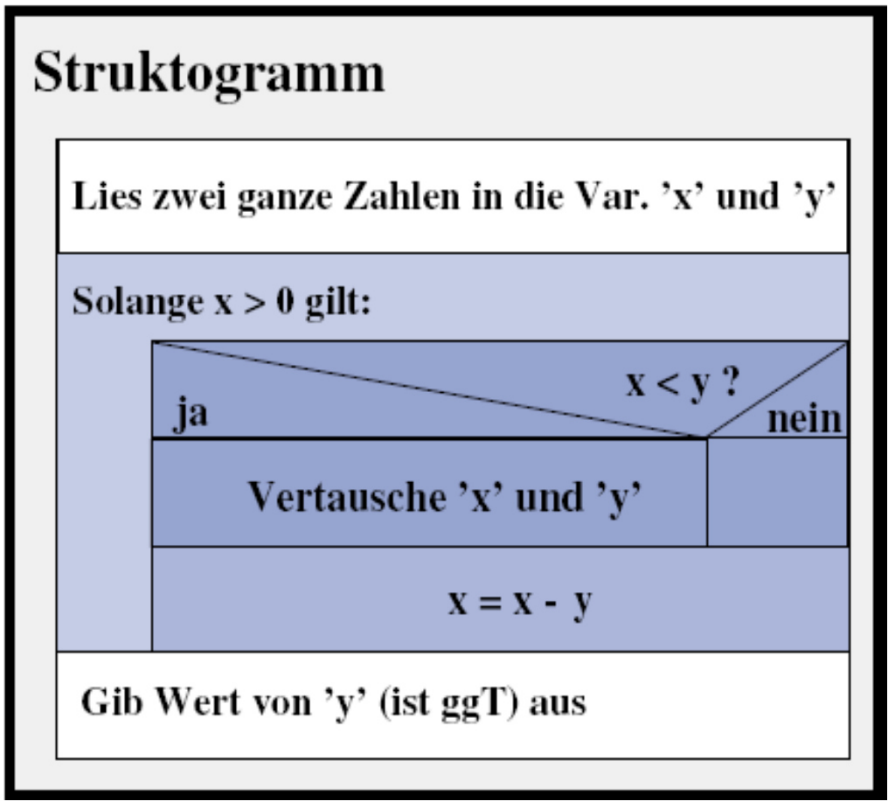
- Programmablaufpläne oder Flussdiagramme (DIN 66001)



Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Formulierung eines Algorithmus

- Struktogramme (von Nassi und Shneidermann)
  DIN 66261



Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Schritt 3: Implementierung des Algorithmus

- Umsetzen eines Algorithmus in die entsprechende Programmiersprache

	Sprachenart	Beispiel	Merkmale (Abstraktion von)
1G	Maschinen	Maschinen-Code	Binäre Befehle
2G	Assembler	Assembler-Code	Symbolische Befehle
3G	prozedural	FORTRAN, COBOL	Hardware-unabhängig
3G+	funktional objektorientiert	PASCAL, C, C++, Java, C#	Strukturiert (Daten=abstrakte Datentypen) Objektorientiert
4G	deklarativ	LISP,PROLOG,SQL	Transaktions-orientiert
5G	???		

Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Sprachen ab der 3. Generation = höhere Programmiersprachen

Schritt 3: Implementierung des Algorithmus

- Ergebnis: Programm
  - Besteht in der allgemeinsten Form aus:
    1. Objekten (Daten)
    2. Algorithmus, der Operationen an den Objekten (Daten) festlegt
  - Verfahren, das gegebene Daten durch einen Algorithmus in einem Computer manipuliert, um ein bestimmtes Endergebnis (Endzustand) zu erreichen
  - Satz von Befehlen, der zur Ausführung einer bestimmten Aufgabe zusammengestellt wurde

Rezept als „Programm“

```
// definiere Namen für Zutaten („Variablen“)
t_gemuese kar, sel, kr;
t_fleisch fl;
t_fluessigkeit oel, wein, bruehe;
t_pulver staerke;
t_gewuerze sl, pf;
```

```
// belege Variablen mit Werten
kar.art = karotte;
kar.gewicht_g = 200;

sel.art = knollensellerie;
sel.gewicht_g = 400;

kr.art = kohlrabi;
kr.gewicht_g = 300;
```

```
// belege Variablen mit Werten
fl.art = rinderbeinscheiben;
fl.anzahl = 4;
fl.gewicht_g_pro_stueck = 250;
```

Rezept als „Programm“

```
// belege Variablen mit Werten
oel.art = sonnenblumen;
oel.vol_ml = 45;

wein.art = rot;
wein.rebsorte = spaetburgunder;
wein.geschmacksgrad = trocken;
wein.vol_ml = 250;
wein.alk_prozent_min = -1;  // -1 = egal
wein.alk_prozent_max = -1;
wein.jahrgang = -1;
wein.winzer = -1;
wein.land = D;

bruehe.art = rind;
bruehe.vol_ml = 250;
```

Rezept als „Programm“

```
// belege Variablen mit Werten
staerke.art = speisestärke;
staerke.anz_tl = 1.5;

sl.art = Salz;
sl.gewicht_g = 5;

pf.art = pfeffer;
pf.gewicht_g = 5;
```

Rezept als „Programm“

```
// nun der „Programmcode“
// „putze()“ muss als Funktion extra programmiert werden
// und an dieser Stelle bekannt sein
// ebenso die anderen Funktionen
putze(kar);
putze(sel);
putze(kr);

wasche(kar);
wasche(sel);
wasche(kr);

schaele(kar);
schaele(sel);
schaele(kr);

 //definiert als schneide(t_gemuese, t_art, t_breite_mm)
schneide(kar, scheiben, 5);
```

Rezept als „Programm“

```
//definiert als schneide(t_gemuese, t_art, t_breite_mm)

schneide(kar, scheiben, 5);
schneide(sel, wuerfel, 10);
schneide(kr,  wuerfel, 10);

wasche(fl);
trockne(fl);

braeter_auf_herdplatte_stellen();
herdplatte_schalter(5);  // Schalterstellung am Herd = 5
warte_bis_temp_erreicht(200);  // 200°c in Bräter

gib_in_braeter(oel);
warte_s(10);          // warte 10 Sekunden
gib_in_braeter(fl);
warte_s(120);         // brate Fleisch 120 Sekunden
wende(fl);            // Fleisch wenden
warte_s(120);
```

Rezept als „Programm“

```
gib_in_braeter(kar);
gib_in_braeter(sel);
gib_in_braeter(kr);

warte_s(120);

gib_in_braeter(wein);
gib_in_braeter(bruehe);
herdplatte_schalter(9);          // Herd auf Vollgas
warte_bis_temp_erreicht(98);     // es kocht

herdplatte_schalter(2.5);        // so, dass es köchelt

warte_s(1800);                   // 30 Min.


usw…
```

Schritt 4: Übersetzung des Programms

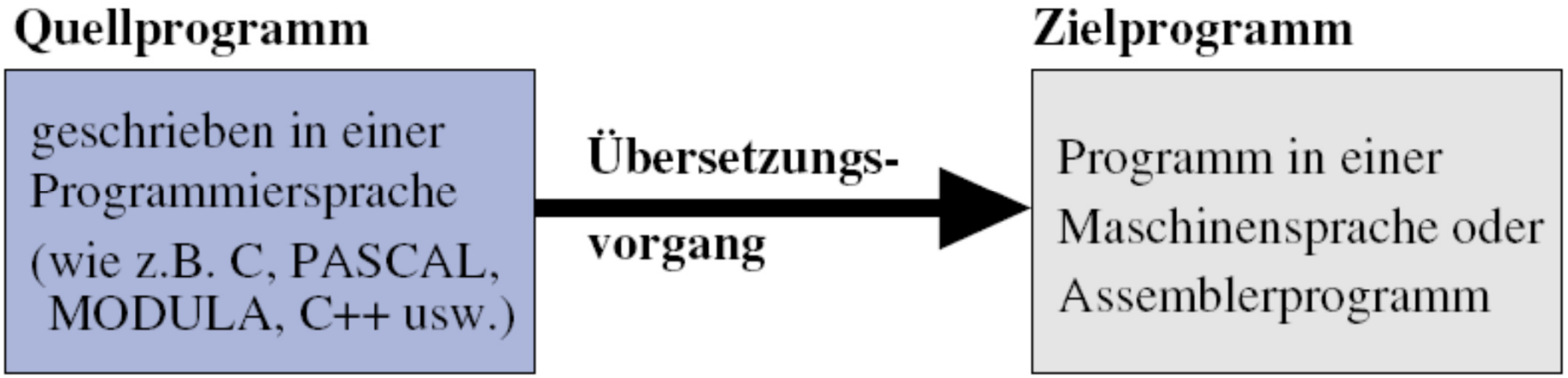
- Zwei verschiedene Möglichkeiten für die Übersetzung vom Programm zum Maschinenprogramm:
  - Compilieren
    - Jeder Befehl des in der höheren Programmiersprache geschriebenen Programms wird vor dem Ablauf in eine entsprechende Folge von Maschinenbefehle übersetzt und in einer Datei gespeichert
    - Anschließend Ausführung des Maschinenprogramms aus dieser Datei
    - Programm, das für die Übersetzung zuständig ist, wird Compiler genannt

Schritt 4: Übersetzung des Programms

- Interpretieren
  - Jeder Befehl des Programms wird nach Prüfung und Dekodierung unmittelbar ausgeführt
  - Es wird kein auf einmal in Maschinensprache übersetztes Programm generiert und abgespeichert
  - Programm, das für die Übersetzung und Ausführung der einzelnen Befehl zuständig ist, wird Interpreter genannt

Compiler

- Werkzeug, welches das von einem Programmierer erstelltes Programm (*Quellprogramm* oder *Source Code)* in Maschinensprache umwandelt



Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

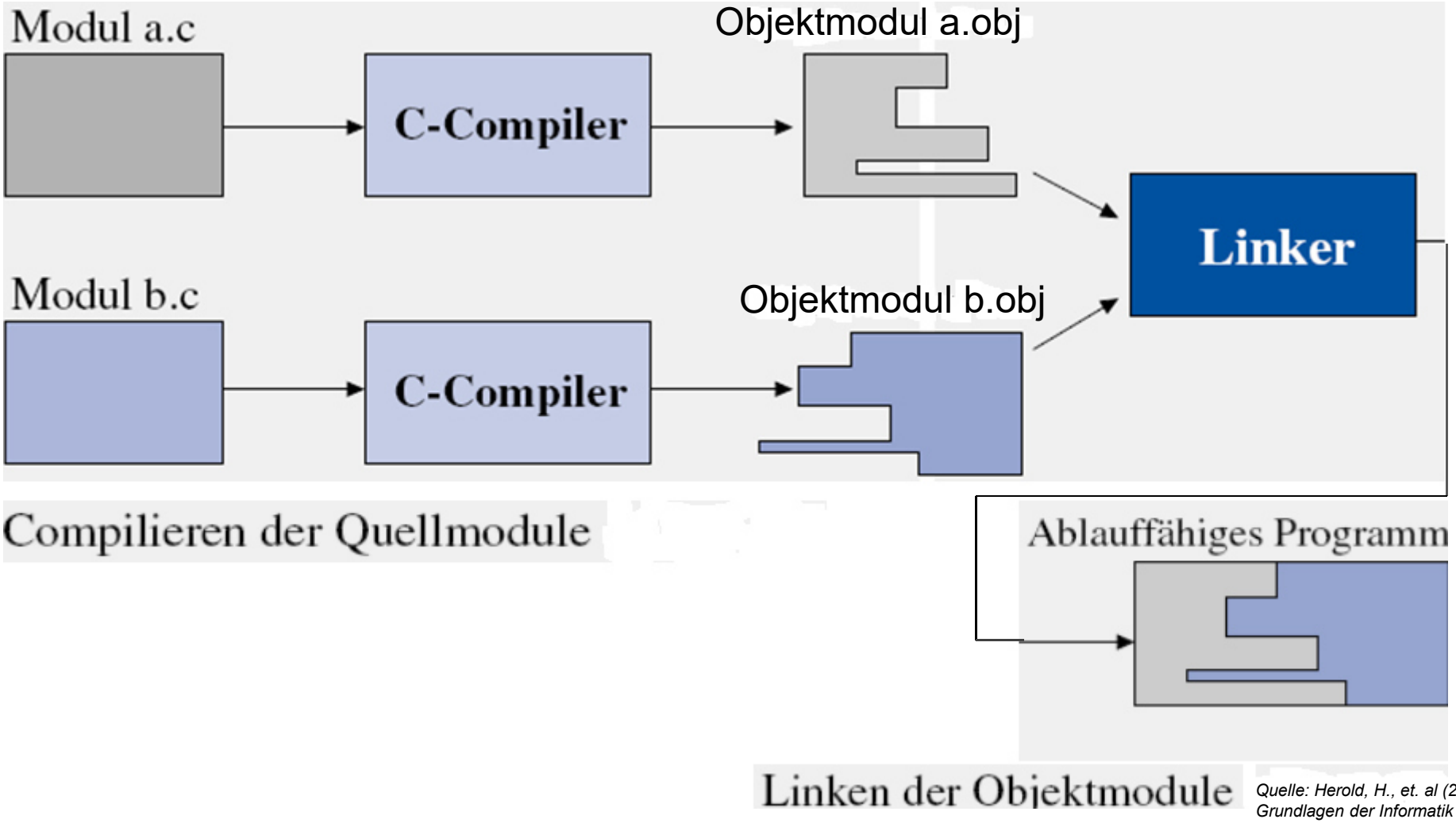
Schritt 5: Linken

- Grundidee der Software-Entwicklung:
  getrennte Compilierung
  - Aufgabenstellung wird in kleine, überschaubare Teilaufgaben zerlegt
  - Jede einzelne Teilaufgabe wird als eigenständiges Programmteil (= Modul) realisiert
  - Unterschiedliche Module können zu unterschiedlichen Zeitpunkten von verschiedenen SW-Entwicklern unabhängig voneinander compiliert werden
  - Resultierende Objektdateien werden mit dem Linker zu einem ablauffähigen Programm zusammengebunden

Linker (1)

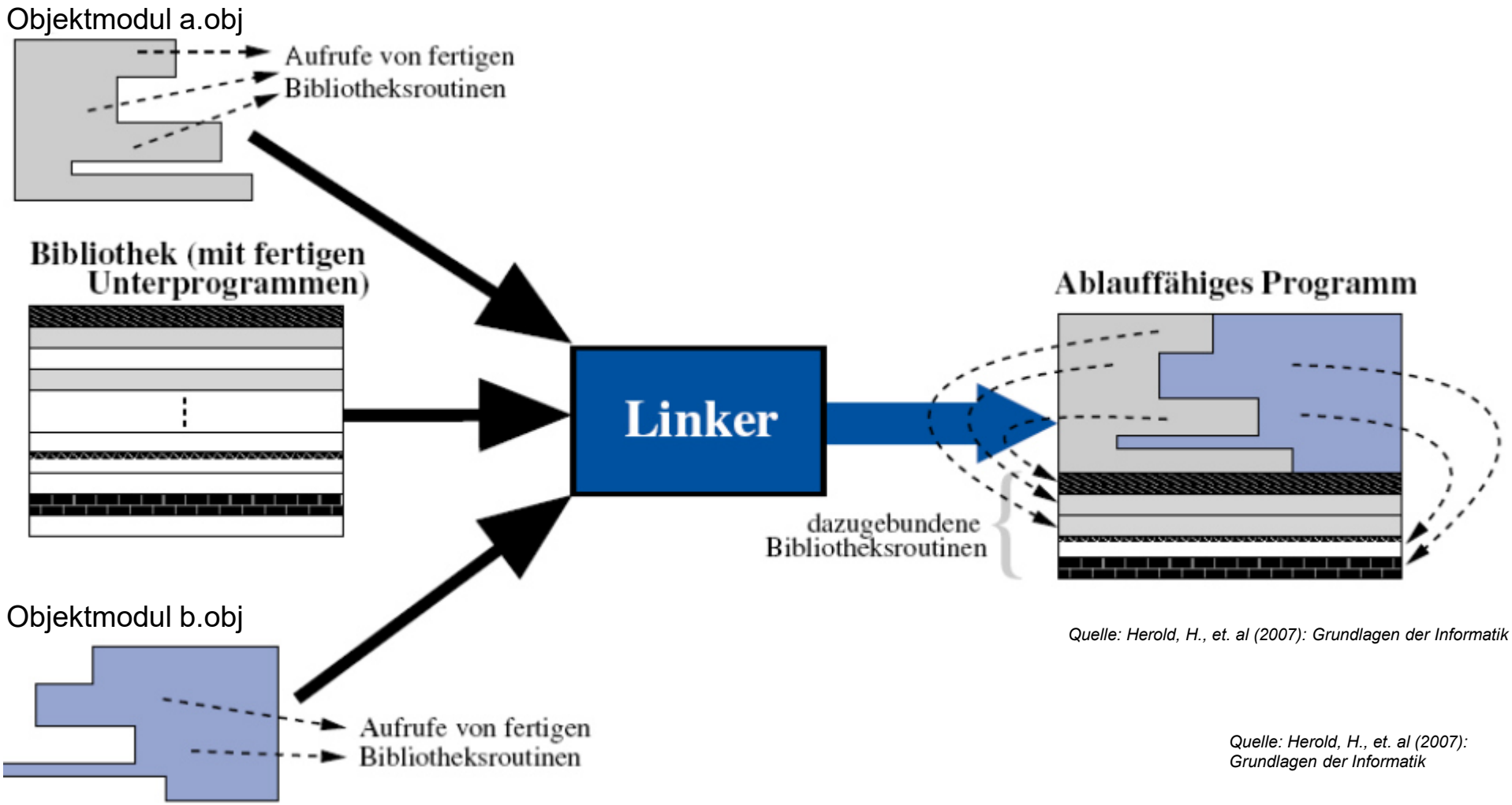
- Normalerweise ein eigenes Programm,
  - das unabhängig vom Compiler ist und die vom Compiler übersetzten Objektmodule erhält
  - Ggf. nach fertigen Bibliotheksroutinen in den entsprechenden Bibliotheken sucht (Modulaufrufen)
  - das die resultierenden Dateien zu einem ablauffähigen Programm zusammenbindet

Linker (2)



Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Linker (3)



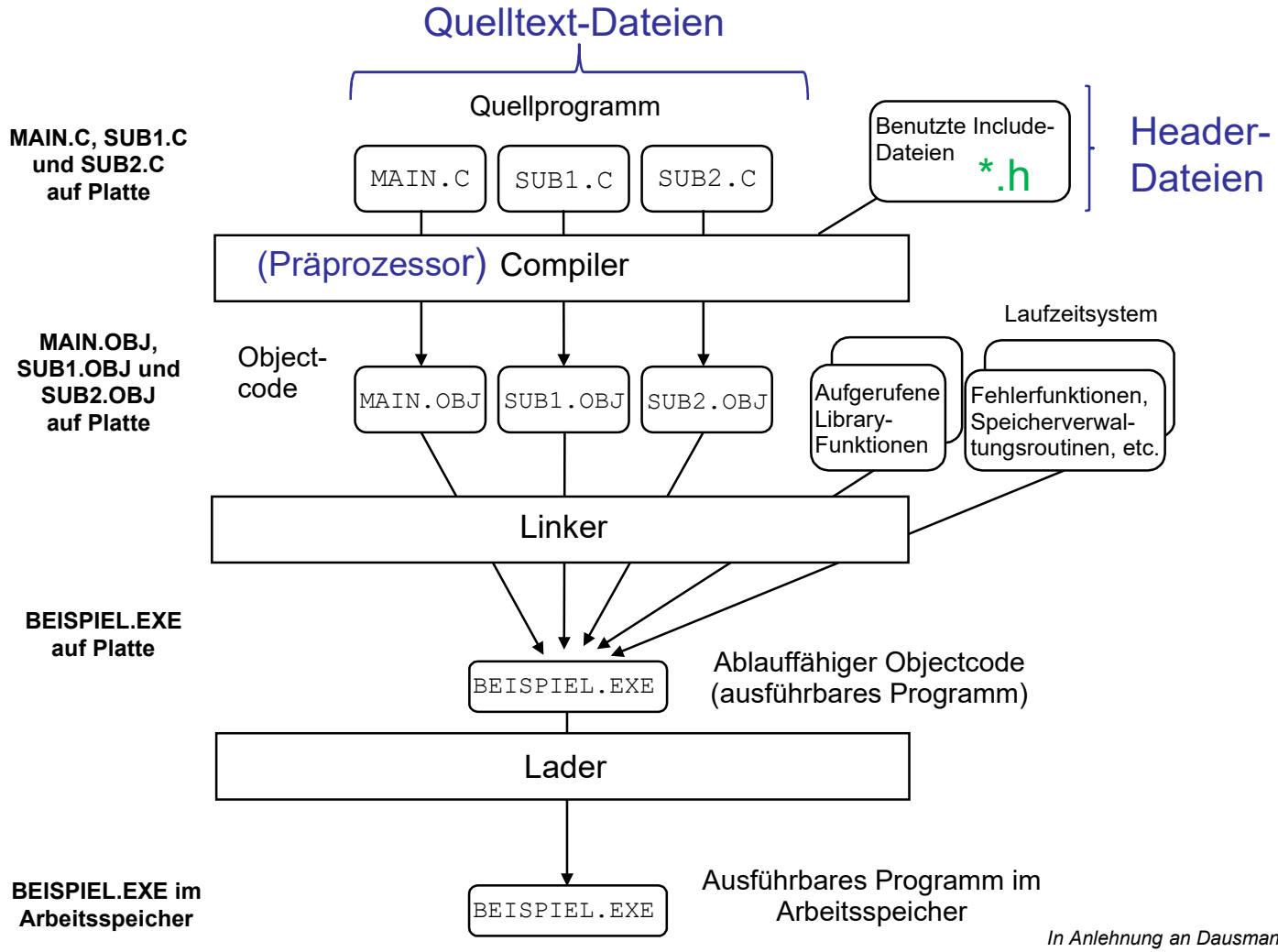
Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Quelle: Herold, H., et. al (2007): Grundlagen der Informatik

Schritt 6: Starten des Programms

- Lader
  - Liest ein gelinktes Programm Befehl für Befehl ein und trägt den Maschinencode jeweils an die Stelle im Hauptspeicher ein, die als Speicheradresse vor jedem Befehl bzw. beim Programmstart angegeben wurde

Ablauf Programmerstellung und -ausführung



In Anlehnung an Dausmann, et.al (2008). S. 33

Prozess der Programmerstellung und -ausführung (1)

1. Erstellung der Programmdateien
   - Mit einem Editor (= Werkzeug zur Texterstellung)
   - 2 Arten von Dateien:

     - **Quelltext**-Dateien (Endung: .c)
       - Enthalten eigentlichen Programmtext
       - Jede Datei ist eine in sich abgeschlossene, kompilierfähige Einheit mit einer definierten Funktionalität

     - **Header**-Dateien (Endung: .h)
       - Beinhalten Bekanntmachungen (Deklarationen) und Informationen über den strukturellen Aufbau von neu definierten Datentypen
       - Werden zur Übersetzung von Quelltext-Dateien benötigt

Prozess der Programmerstellung und -ausführung (2)

2. Übersetzung/Kompilierung der Programmdateien
   (1) Starten des Präprozessors
   - Einfacher Textersetzer
   - Durchsuchen des Quelltexts nach Präprozessor-Anweisungen und Ausführung dieser Anweisungen
   - Modifikation des Programmtextes

   (2) Syntaxüberprüfung
   - Kontrolle, ob der Text den formalen Regeln der Programmiersprache C entspricht

   (3) Übersetzung der C-Befehle in Assemblersprache
   - Textuelles Pendant zu Maschinensprache
   - Verwendung verschiedener Assembler möglich

   (4) Umwandlung des Programms in die Maschinensprache
   - Assembler erzeugt die Objektdatei (Endung: .obj)

Prozess der Programmerstellung und -ausführung (3)

3. Binden/Kombinieren aller Objektdateien plus entsprechender Bibliotheken bzw. Module
   - Linker erzeugt ausführbare Datei (Endung: .exe)
   - Standard-Bibliothek oder selbst erstellte, schon früher übersetzte Programmteile
4. Lader lädt ausführbare Datei in Arbeitsspeicher

Quellcode → Präprozessor

Originalcode (komplett)

```c
#include <stdio.h>  // Standard Input/ Output  z.B. scanf, printf
#define TEXT "Dies ist ein Text\n"

int main(void)
{
  const float zahl = 5.2f;  // definiert eine Konstante

  printf ("Zahl: %10.2f\n", zahl);  /* Ausgabe */
  printf(TEXT);

  return(0);
}
```

nach Präprozessor
(kleiner Ausschnitt)

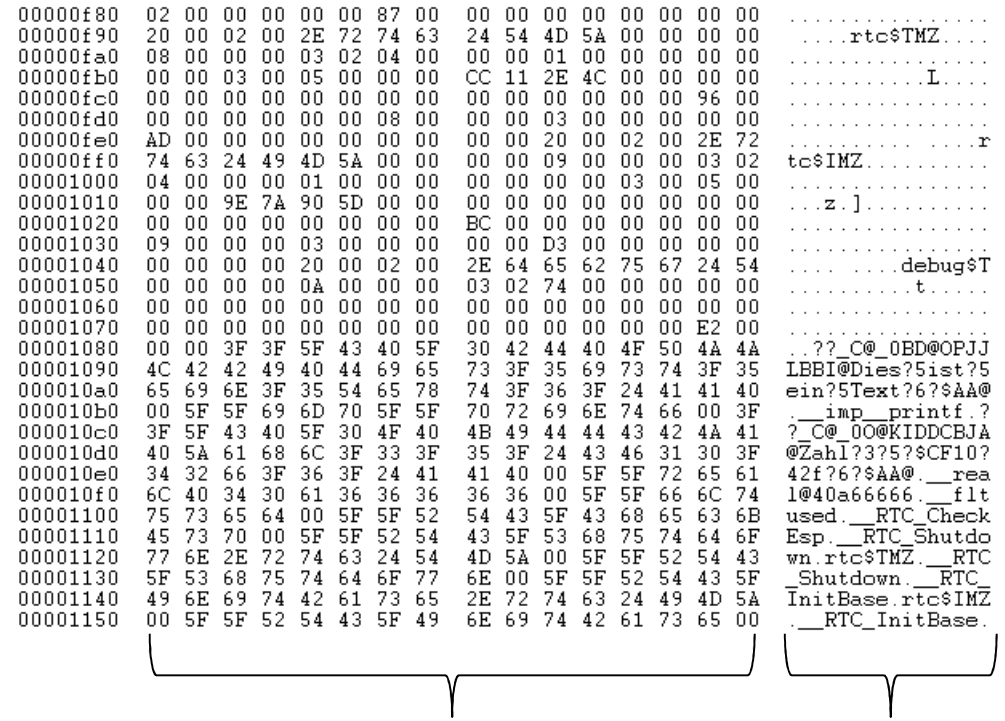
```c
…
#line 283 "c:\\program files (x86)\\microsoft visual studio 10.0\\vc\\include\\stdio.h"
 __declspec(dllimport) int __cdecl _pclose(   FILE * _File);
 __declspec(dllimport) FILE * __cdecl _popen(   const char * _Command,    const char * _Mode);
 __declspec(dllimport) int __cdecl printf(    const char * _Format, ...);
…

int main(void)
{
  const float zahl = 5.2f;
  printf ("Zahl: %10.2f\n", zahl);
  printf("Dies ist ein Text\n");

  return(0);
}
```

Assembler-Code (Ausschnitt)

```
_TEXTSEGMENT
_zahl$ = -8; size = 4
_mainPROC; COMDAT
; Line 5
pushebp
movebp, esp
subesp, 204; 000000ccH
pushebx
pushesi
pushedi
leaedi, DWORD PTR [ebp-204]
movecx, 51; 00000033H
moveax, -858993460; ccccccccH
rep stosd
; Line 6
fldDWORD PTR __real@40a66666
fstpDWORD PTR _zahl$[ebp]
; Line 8
fldDWORD PTR _zahl$[ebp]
movesi, esp
subesp, 8
fstpQWORD PTR [esp]
pushOFFSET
??_C@_0O@KIDDCBJA@Zahl?3?5?$CF10?42f?6?$AA@
callDWORD PTR __imp__printf
addesp, 12; 0000000cH
cmpesi, esp
call__RTC_CheckEsp
; Line 9
movesi, esp
pushOFFSET
??_C@_0BD@OPJJLBBI@Dies?5ist?5ein?5Text?6?$AA@
callDWORD PTR __imp__printf
addesp, 4
cmpesi, esp
call__RTC_CheckEsp
; Line 11
xoreax, eax
; Line 12
popedi
popesi
popebx
addesp, 204; 000000ccH
cmpebp, esp
call__RTC_CheckEsp
movesp, ebp
popebp
ret0
_mainENDP
_TEXTENDS
```

Objektdatei (Ausschnitt)

Zeilennummer
(nicht Bestandteil der Datei)

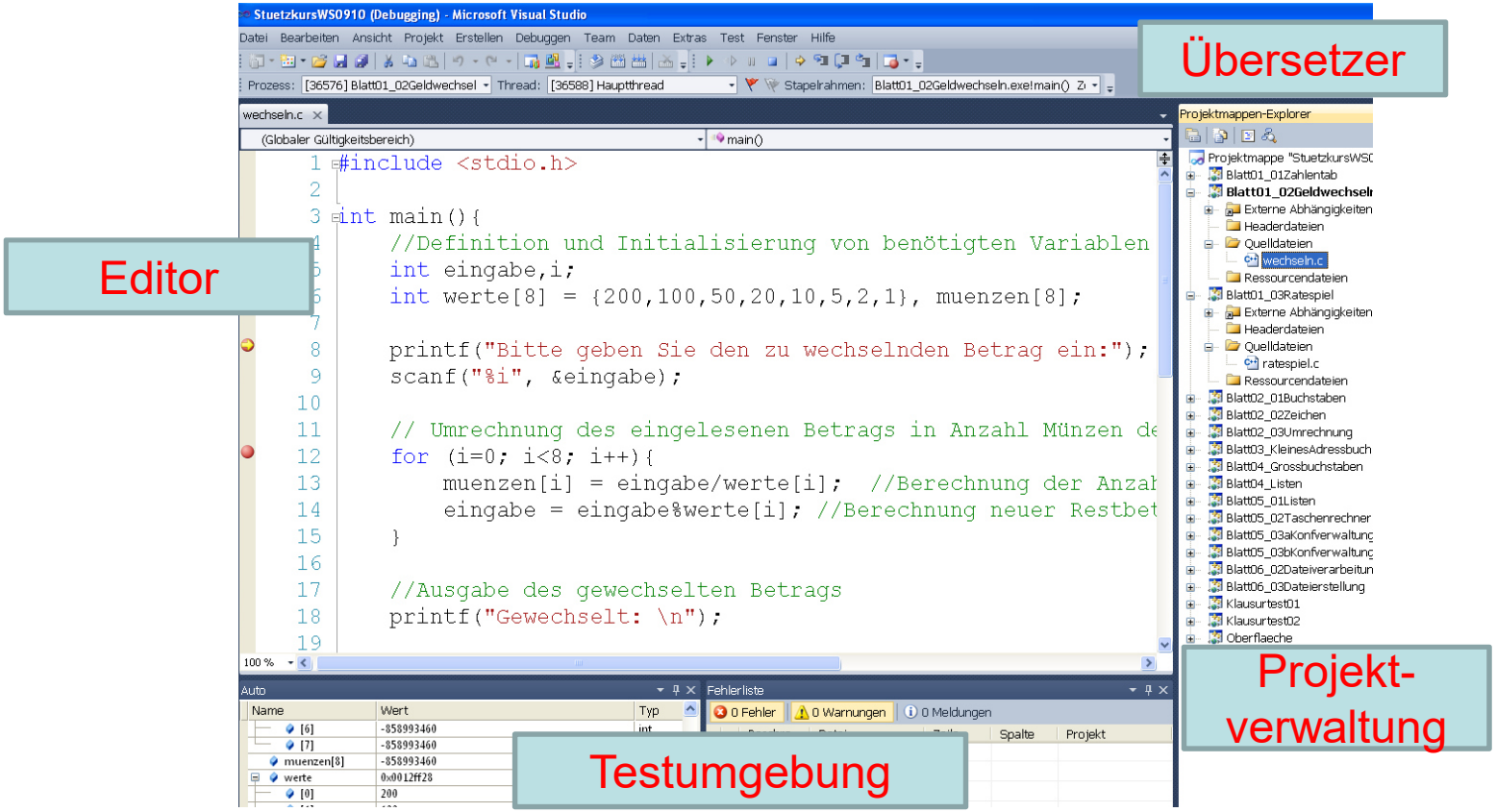


Ausführbare Datei (Ausschnitt)

```
0000000000000000000000000000000000000¢ppN0000B000j000ÀcA0ÀM00Dies
 ist ein TextJ000000Zahl: %10.2fJ000ff¦@f0:0\0d0d0\0v0c0t0o0o0l0
s0\0c0r0t0_0b0l0d0\0s0e0l0f0_0x0806O\0c0r0t0\0s0r0c0\0c0r0t0e0x0
e0.0c0000000000000000000000000000_0_0n0a0t0i0v0e0_0s0t0a0r0t0u0p0

 ...
00000000D00000A0X000X00€00000000D00000A0A000000€00000000D00000A0
>D00H000X A0ZA00äD000000<assembly xmlns="urn:schemas-microsoft-c
om:asm.v1" manifestVersion="1.0">MJ  <trustInfo xmlns="urn:schem
as-microsoft-com:asm.v3">MJ    <security>MJ      <requestedPrivi
leges>MJ        <requestedExecutionLevel level="asInvoker" uiAcc
ess="false"></requestedExecutionLevel>MJ      </requestedPrivile
ges>MJ    </security>MJ  </trustInfo>MJ</assembly>PAPADDINGXXPAD
DINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPADDINGPADDINGXXPAD

0PA0Ø000 3³3¹3Ê3Ð3T4"656‹6"6ž6£6¨6-6³6¹6À6Æ6Ø6à6æ6ô6ÿ667C7H7M7S7
Y7^7c7i7q7w7È7Í7ß7÷7H83898^8f8{8„8 8'8¼8Æ8Î8Ó8á8ë8ó8ÿ8G9]9#9)919
I9Q9Z9`9g9n9u9ƒ9‰9'9 ˜9ž9§9Ö9Ü9å9ì9ò9û9G:Û:p;w;';Ü;+<r<-<Ñ<[=;=`=
'=¹=x>}> >Î>(?-??? ?ü?000 A0ð000G0R0Œ0À0Å0<1H1d1j1w1~1ƒ1ž1¤1-1¶1
»1Æ1æ1ï1ø1E2S2B2ì2ó2L3W3'3Ì3O4E4 4³4°4Ù4ß4å4>5O5U515A5H5Ö5Ü5ì5ú5
ÿ5D6t6¤6«6¾6Õ6Û6î6H7M7¶737:7M7l7s7{7,7—7-7²7¾7Á7V8\8b8^8˜8_8¨8·8
Ì8Ø8ä8ô8=9H9Š9 9-9œ9¢9Ø:Ý:ï:H;(<.<4<V<[<§<Ä< =)=6=@=H=M=T=¥=s>z>
ü>C?ü?0000A0¼000G0!0&0,0<0Z0_0,0 0©0¿0ô0S1x1~1 1-1Ÿ1¨1²1Æ1b2t2z2

€2†2Œ2'2 ˜2¬2²2¸2¾2Ä2Ê2Ñ2Ø2ß2æ2í2ô2û2C3K3S3¶3(3-333=3G3S3_3d3v3{3
 3‡3Ÿ3¦3ô3ú3O4F4L4R4X4^4$4*40464<4B4H4N4T4Z4`4f4l4r4x4~4„4Š4 4-4
œ4000PA0 000D1P4T5d;h;l;p;t;x;ø?ü?000`A0(00000D0H0„2^2D6P9l;p;Œ;
 ;¬;°;Ð;ì;ð;0000000000000000000000000000000000000000000000000000
```

Integrierte Entwicklungsumgebung (1)

- Heutzutage werden Software-Projekte in sog. Entwicklungsumgebungen erstellt



Integrierte Entwicklungsumgebung (2)

- Integrierte Entwicklungsumgebungen enthalten
  - Compiler
  - Linker
  - Lader
  - Debugger
  - Editor
  - Projektverwaltungen
- Komfortable Programmentwicklung möglich
- Wir verwenden in den Übungen Microsoft Visual Studio

Debugger

- Programmierwerkzeug, um den Ablauf eines Programms während der Fehleranalyse exakt beobachten zu können (Fehlerlokalisierung)

- Einsatz:
  - Laden von Programmen, die gezielt gestartet werden
  - Anhalten an beliebigen Stellen (Setzen von Haltepunkten)
  - Programme nach dem Anhalten fortsetzen
  - Programm Schritt für Schritt ausführen
  - Anzeigen von Speicherinhalten

Zusammenfassung

- Begriff „Algorithmus“
- Programmerstellung und -ausführung