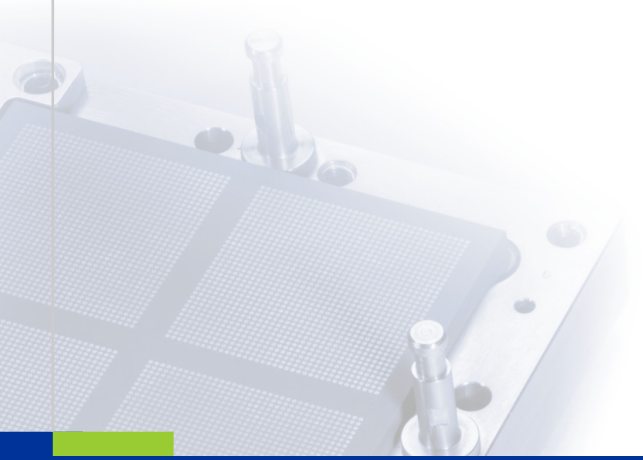
Programmierung mit C

Einführung



Danksagung

Ich danke Prof. Dr.-Ing. Karl-Heinz Niemann (Fachbereich Elektro- und Informationstechnik an Fachhochschule Hannover) für die Korrektur und Ergänzung meiner Folien sowie Überlassung der Druckfolien.

Organisatorisches

- In vielen Programmbeispielen wurden kurze Variablennamen verwendet, um die Beispiele auf den Folien darstellen zu können.
- Für Ihre Programme sollten Sie aussagekräftige Variablennamen wählen.
- In einigen Programmbeispielen befinden sich Zeilenumbrüche, um die Programme auf der Folie darstellen zu können. In Ihren Programmen sollten Sie Anweisungen, wenn möglich, auf eine Zeile beschränken.

Die zur Verfügung gestellten Folien werden durch die Veranstaltung führen, sind jedoch für ein vollständiges Erlernen der Programmiersprache nicht ausreichend.

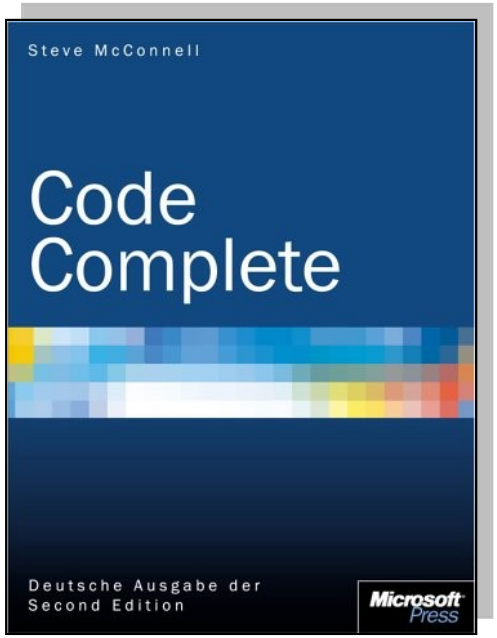
Es wird die zusätzliche Nutzung der folgenden Literatur empfohlen.

- /1/ Skript von Prof Schuppe: **Programmiersprache C.**
- /2/ **Grundlagen der Programmierung**. Buch des Regionalen Rechenzentrums für Niedersachsen.

Literatur

- /3/ **Die Programmiersprache C**. Ein Nachschlagewerk. Buch des Regionalen Rechenzentrums für Niedersachsen.
- /4/ Brian W. Kernighan, Dennis M. Ritchie **Programmieren in C.** Mit dem C-Reference Manual in deutscher Sprache. 2. Ausgabe ANSI C. 284 Seiten, ISBN 3-446-15497-3.
- /5/ Clovis L. Tondo, Scott E. Gimpel **Das C-Lösungsbuch** zu "Kernighan/Ritchie, Programmieren in C". 2. Ausgabe ANSI C. ISBN 3-446-15946-0.

Literatur



/6/ Mc Connell, Steve: **Code Complete.** Deutsche Ausgabe, Second Edition. Microsoft Press, Unterschleißheim, 2005. ISBN 3-86063-593-X. Preis 49,90 €.

- Dieses Buch ist nicht unbedingt für Anfänger gedacht, gibt aber einen sehr guten Überblick über risikoarme Programmierung und guten Programmierstil. Programmbeispiele in Visual Basic, Java und C++.
- Webseite des Autors zum Buch http://www.cc2e.com/ enthält alle Checklisten etc.

Interessante Links zu C sowie Bibliotheken

R|R|Z|N

- C Übersicht mit Programmbeispielen:
  - http://www.pronix.de/pronix-4.html
- Skripte zu C von Prof. Dr. J. Dankert:
  - http://www.haw-hamburg.de/rzbt/dankert/c_tutor.html/
- Lexikon zu C:
  - http://www.prenhall.com/jaeschke/
- C-Reference Manual
  - http://www.cs.bell-labs.com/who/dmr/cman.ps
- Übersicht über die C-Standardbiliothek:
  - http://www.fh-fulda.de/~klingebiel/c-stdlib/index.htm
  - http://www.dinkumware.com/libraries.html
  - http://www.infosys.utas.edu.au/info/documentation/C/CStdLib.html#stdio.h

Hilfreiche Links zum Programmierstiel

- Standards and Style for Coding in ANSI C
  http://www.jetcafe.org/~jim/c-style.html
- Recommended C Style and Coding Standards
  http://www.chris-lott.org/resources/cstyle/indhill-cstyle.html
- how to write unmaintainable code (ein kleiner Spass zum Schluss)
  http://jdj.sys-con.com/read/35819.htm
- Programmierrichtlinien
  http://www.uwe-sauerland.de/richtlinien/Programmierstil.html
- Standards im Automobilbau
  http://www.misra-c2.com/
- GNU Coding Standards
  http://www.gnu.org/prep/standards/

Für die Erstellung der C-Programme können unter Windows folgende Compiler hierzu verwendet werden:

- Mingw C-Compiler (Zugriff auf die Windows-API möglich)
  http://www.mingw.org

- Cygwin C-Compiler (Zugriff auf die Windows- und UNIX-API möglich)
  http://www.cygwin.com

- Borland C-Compiler
  http://www.borland.com/products/downloads/download_cbuilder.html

Für die Betriebssysteme Linux sowohl als auch Windows kann der Compiler genutzt werden:

- Gnu C-Compiler (gcc)
  http://www.gnu.org

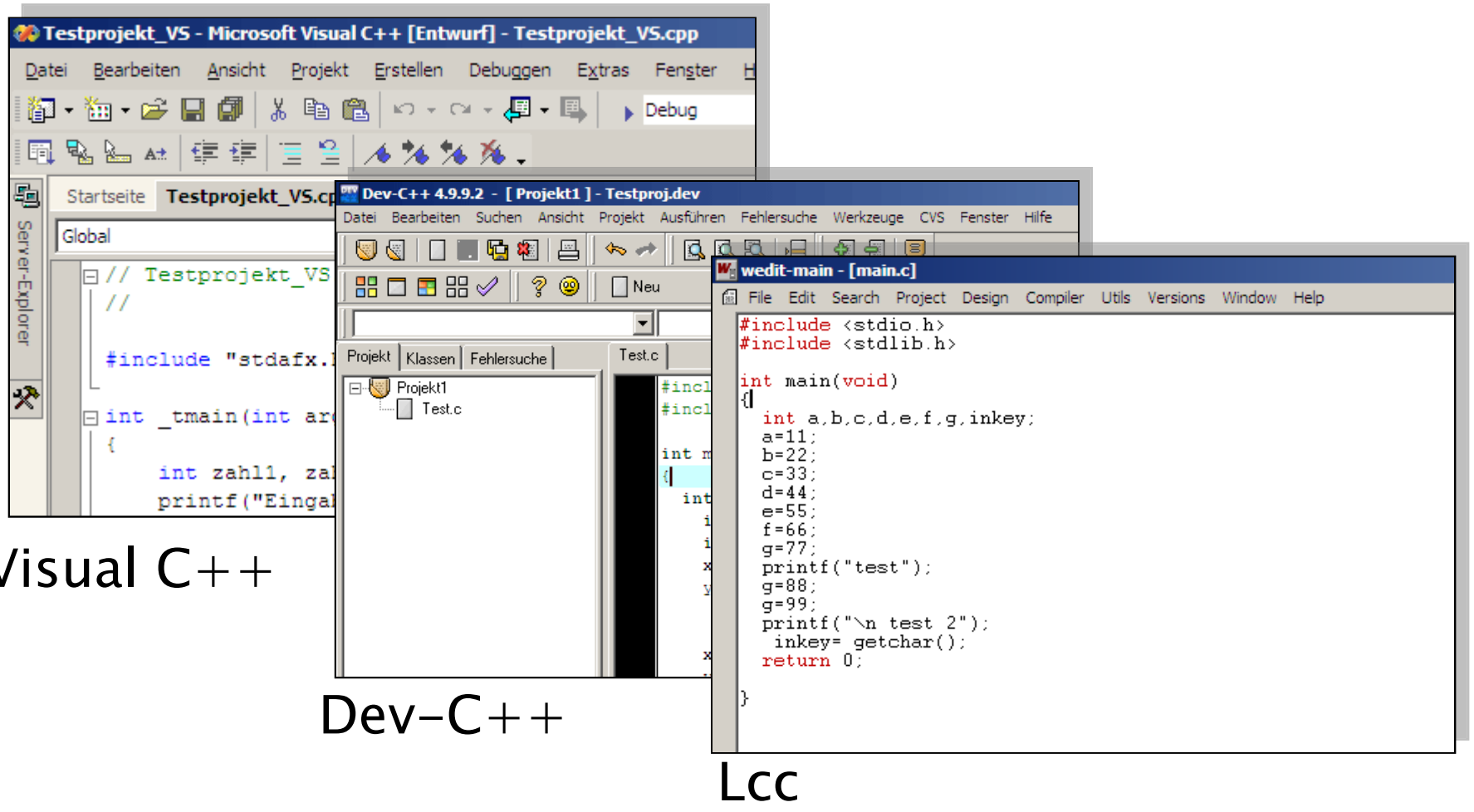
Für die Erstellung der C-Programme können unter Windows folgende Entwicklungssysteme hierzu kostenlos verwendet werden:

- Dev-C++ -System. Download unter:
  http://www.bloodshed.net/devcpp.html
- Lcc-Win32-Compiler. Download unter:
  http://www.cs.virginia.edu/~lcc-win32/
- Pelles C
  http://www.smorgasbordet.com/pellesc/index.htm

Für die Betriebssysteme Linux sowohl als auch Windows kann die Entwicklungsumgebung kann folgende IDE genutzt werden:

- Eclipse mit dem Plug-In CDT
  http://www.eclipse.org/
- Code::Blocks (auch für MAC)
  http://www.codeblocks.org/

Entwicklungssysteme



Inhaltsübersicht Teil 1

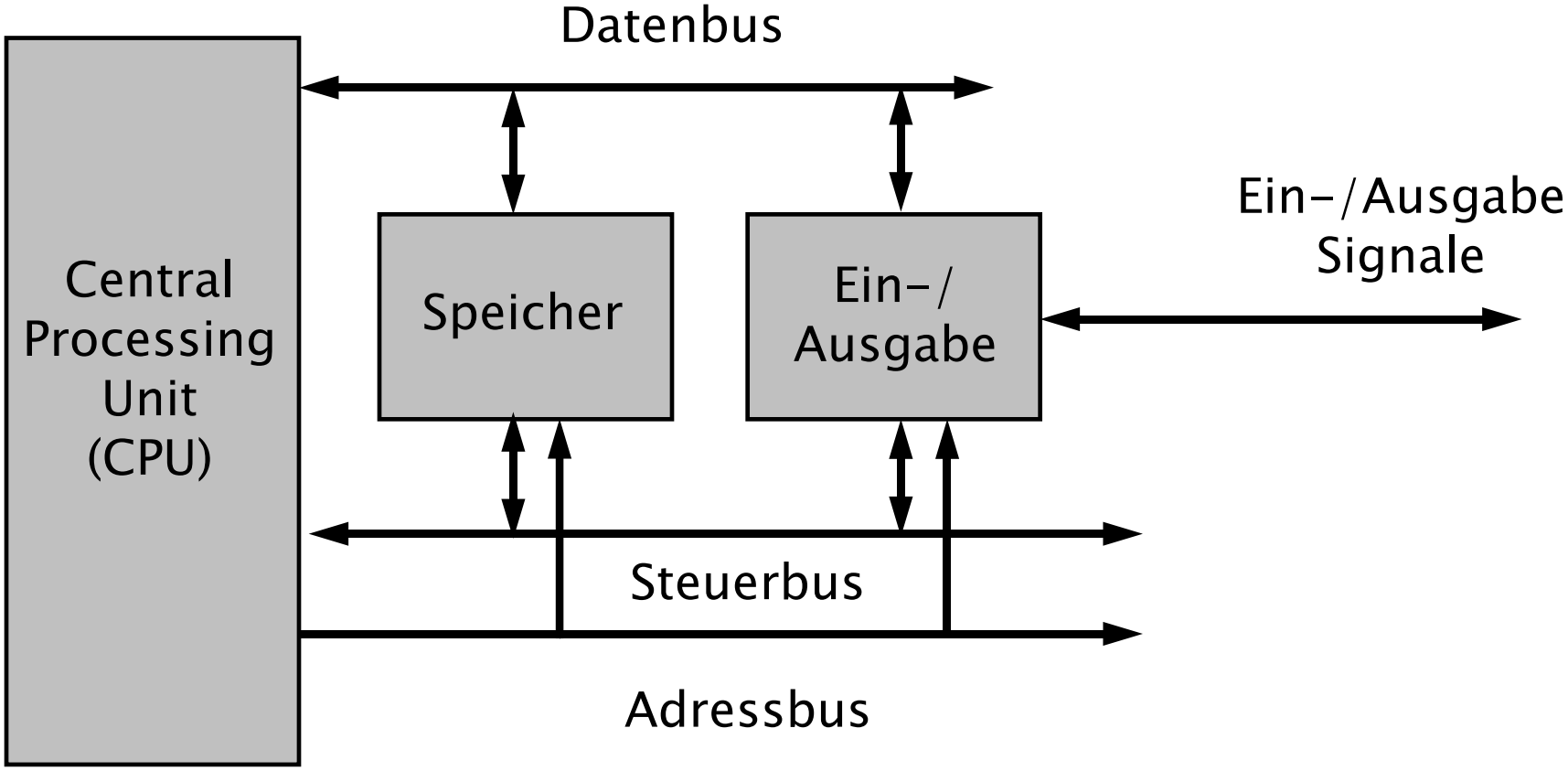
R|R|Z|N

- Funktionen eines Rechners.
- Was ist ein Algorithmus?
- Wie entsteht aus einem Algorithmus ein Programm?
- Aufgaben eines Compilers und Präprozessors.
- Informationen zur Programmiersprache C.
- Wie ist ein C-Programm aufgebaut?
- C-Programme kompilieren und ausführen.

Prinzipielle Funktionen eines Rechners

Prinzipieller Rechneraufbau

- Der grundsätzliche Aufbau eines Rechners geht auf die Arbeiten von Burks, Goldsteine und von Neumann zurück.
- Nach Ihrer Definition besteht ein Rechner aus den folgenden Grundbestandteilen:
  - Zentraleinheit / Mikroprozessor (engl. Central Processing Unit, kurz CPU genannt)
  - Speicher
  - Ein-/Ausgabeeinheiten


Datenbus
Central Processing Unit (CPU)
Speicher
Ein-/ Ausgabe
Ein-/Ausgabe Signale
Steuerbus
Adressbus

Grundeinheiten eines Rechners

- Die CPU
  - … übernimmt die Ausführung von Befehlen und die zugehörige Ablaufsteuerung.
  - … stellt die die zentrale Recheneinheit im Computer dar.
- Die Komponenten werden durch so genannte Busse miteinander verbunden.
- Busse bezeichnen die Bündelung von mehreren gleichartigen Signalen.
  - Die Dateibusse sind Leitungen zum Transfer von Speicherzelleninhalt. Die Dateibusse sind üblicherweise in 8, 16, 32 oder 64 Bit Breite ausgeführt.
  - Die Adressierung der Speicherzellen erfolgt über den Adressbus.
  - Der Steuerbus fasst alle Signale zusammen, die der externen Steuerung eines Datentransfers dienen.

Grundeinheiten eines Rechners

- Im Speicher werden Daten und Programme abgelegt. Der Speicher selber ist in gleich große Zellen aufgeteilt, um eine lineare Adressierbarkeit zu ermöglichen.
- Die Ein-/Ausgabeeinheit stellt die Verbindung zur Außenwelt her. Über diese Schnittstelle werden Programme sowie Daten ein- und ausgegeben.

- Die Struktur des Rechners ist unabhängig von einem speziellen zu bearbeitenden Problem.
- Es wird für jede Aufgabenstellung ein entsprechendes Programm im Speicher abgelegt, welches dem Rechner sagt, wie er sich zu verhalten hat.
- Dieser Aspekt hat zu der Bezeichnung programmgesteuerter Universalrechner (engl.: stored-program machine) geführt.

Unterscheidungsmerkmale

- War in den Anfängen des Rechnerbaus die CPU noch aus Röhren, später dann diskreten Transistoren aufgebaut, so ist heute die Funktion der CPU in einem einzigen Halbleiterchip integriert. Dieser enthält das vollständige Steuer- und Rechenwerk einer CPU.
- Man unterscheidet Mikroprozessoren hauptsächlich nach:
  - Arbeitsgeschwindigkeit
  - Befehlsvorrat
  - Befehlsformat
  - Wortlänge, d.h. die Zahl der gleichzeitig zu verarbeitenden Bits
  - Adressraum, d.h. die Zahl der adressierbaren Speicherzellen

Anwendungsgebiete

- Neben der eigentlichen Funktion in der Computertechnik finden Mikroprozessoren heute auch Anwendung in sogenannten “Embedded Applications“. Dies sind z. B.:
  - Haushaltsgeräte (Waschmaschinen)
  - Steuergeräte im Automobilbau (z. B. ABS, Einspritzanlage)
  - Elektronische und optische Geräte (z. B. Kameras, DVD-Player, etc.)
  - Kinderspielzeug (Spielkonsolen)
  - Industriesteuerungen (Speicherprogrammierbare Steuerungen, CNC-Steuerungen)

Charakteristika des v. Neumann Rechners

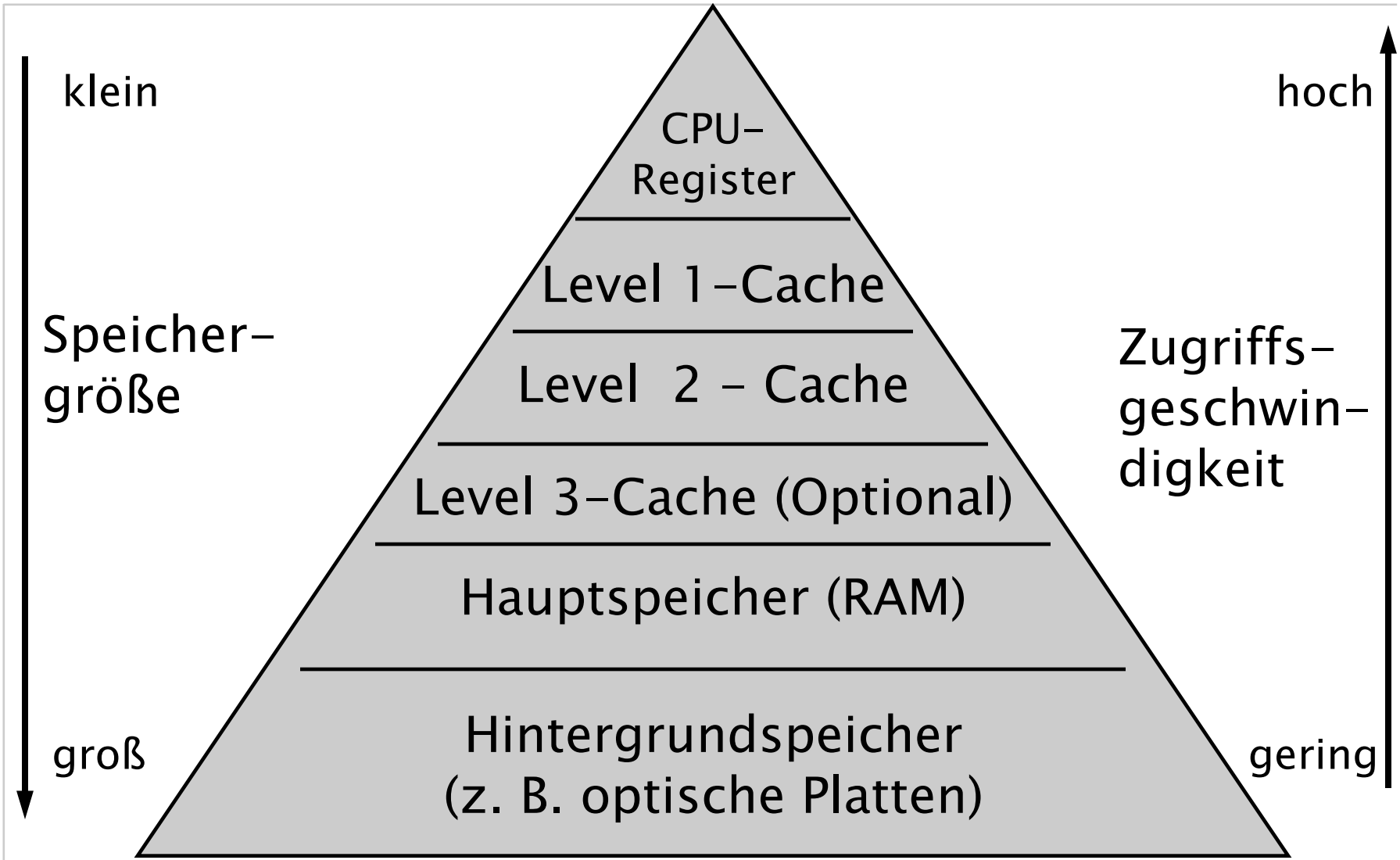
R|R|Z|N

- Zu jedem Zeitpunkt führt die CPU genau einen Befehl aus und dieser kann höchstens einen Datenwert bearbeiten. (Diese Arbeitsweise wird auch als Single Instruction Single Data bezeichnet)
- Alle Speicherworte (d. h. Inhalte der Speicherzellen) sind als Daten, Befehle oder Adressen brauchbar. Die jeweilige Verwendung richtet sich nach dem momentanen Kontext.

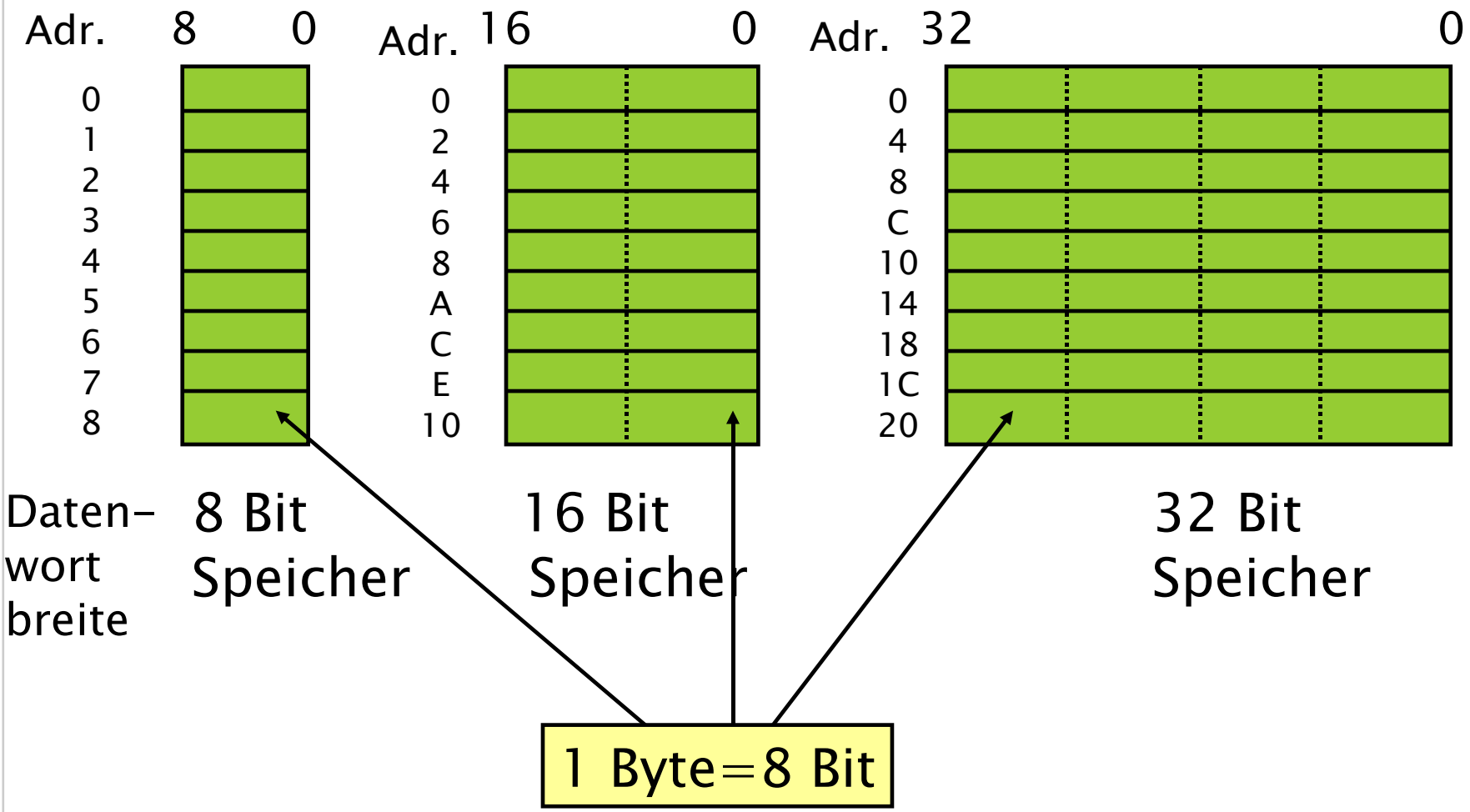
Programmabarbeitung

- Die Programmanweisungen, die Befehle, liegen zusammen mit den Programmdaten im Speicher des Computers.
- Die CPU führt jede Instruktion als Abfolge von Schritten aus. Die Schritte sind vereinfacht die Folgenden:
  1. Hole die nächste Instruktion aus dem Speicher
  2. Ändere den Befehlszähler, damit er auf die nächste Instruktion zeigt
  3. Bestimme den Typ der gerade eingelesenen Instruktion
  4. Wenn die Instruktion ein Wort im Speicher benutzt, bestimme die Position des Wortes
  5. Lies das Wort in ein CPU-Register ein
  6. Führe die Instruktion aus
  7. Gehe zu Schritt 1 und beginne die Ausführung der nächsten Instruktion

Speicherhierarchie



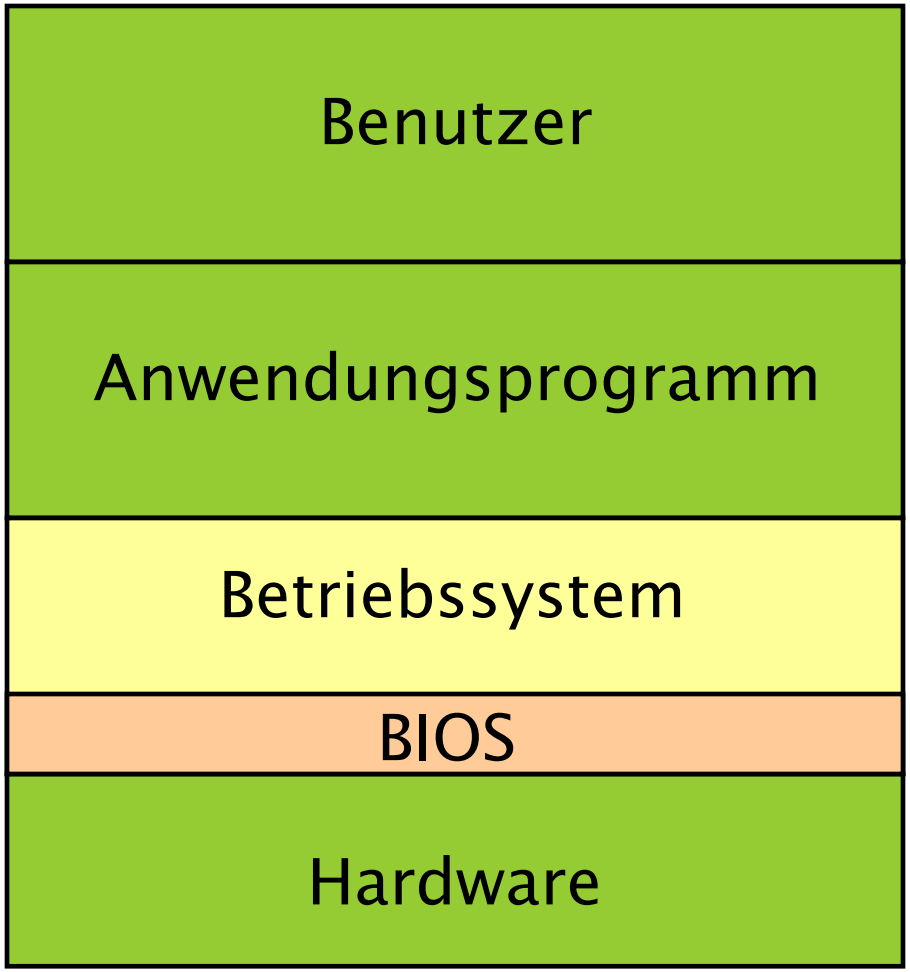
- Der von Neumann Rechner benötigt für die Befehlsausführung Befehle und Programmdaten aus dem Speicher.
- Die Geschwindigkeit der Befehlsausführung ist wesentlich von der Speicherzugriffsgeschwindigkeit abhängig.
- Durch ein mehrstufiges Speicherkonzept werden mehrere Speicherebenen hierarchisch übereinander angeordnet.
- Je dichter der der Speicher an der CPU ist, um so schneller kann auf Ihn zugegriffen werden, um so höher ist sein Preis.
- Je dichter der Speicher an der CPU ist, um so kleiner ist er.
- Der Rechner ist so aufgebaut, dass er Daten in einem möglichst nahe liegenden Speicher hält, und erst auf einen weiter weg liegenden Speicher ausweicht, wenn die Daten im nahen Speicher nicht gehalten / gefunden werden können.


Adr.
8
0
0
1
2
3
4
5
6
7
8
Adr.
16
0
0
2
4
6
8
A
C
E
10
Adr.
32
0
0
4
8
C
10
14
18
1C
20
Daten-
wort
breite
8 Bit
Speicher
16 Bit
Speicher
32 Bit
Speicher
1 Byte=8 Bit

Grundlegende Aufgaben Betriebssystem

- Verbergen der Komplexität der Maschine vor dem Anwender (Abstraktion)
- Bereitstellen einer Benutzerschnittstelle ("Kommandointerpreter", "Shell")
- Bereitstellen einer normierten Programmierschnittstelle (API) für Applikationsprogramme, ggf. auch Compiler, Linker, Editor
- Verwaltung der Ressourcen der Maschine
  - Prozessor(en)
  - Hauptspeicher
  - Hintergrundspeicher (Platte, Band, etc.)
  - Geräte (Bildschirm, Tastatur, Drucker, Plotter, etc.)
  - Rechenzeit
- Verfolgung von Schutzstrategien bei dieser Ressourcenbereitstellung
- Koordination von Prozessen

Stellung des Betriebssystems



Betriebssystemaufrufe

R|R|Z|N

- Die Schnittstelle zwischen dem Betriebssystem und dem Nutzerprogramm wird durch eine Menge von Systemaufrufen (engl. System Calls) gebildet.
- Die Aufrufe unterscheiden sich, je nach Betriebssystem, führen in den verschiedenen Betriebssystemen jedoch ähnliche Funktionen aus.

Beispiele für Betriebssystemaufrufe (1)

R|R|Z|N

Eingabe / Ausgabe

- Daten Ein-/Ausgabe auf Festplatte
- Daten Ausgabe auf Bildschirm
- Daten einlesen von Tastatur

Dateimanagement

- Datei erzeugen
- Datei öffnen
- Datei lesen / schreiben
- Verzeichnis erzeugen oder löschen
- Arbeitverzeichnis wechseln
- Änderung von Dateirechten und Attributen

Beispiele für Betriebssystemaufrufe (2)

R | R | Z | N

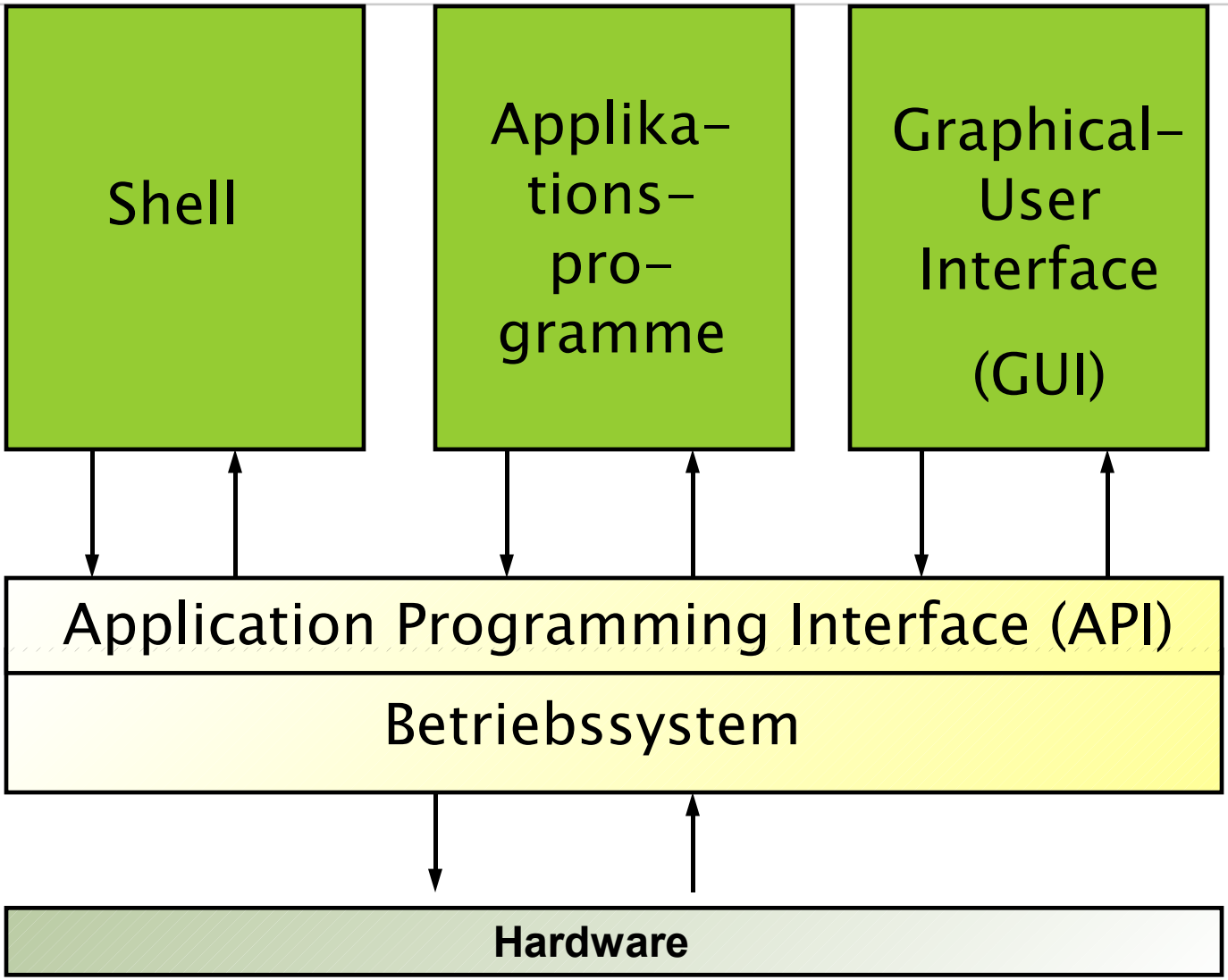
Speicherverwaltung

- Speicher anfordern
- Speicher freigeben
- Speicher reorganisieren

Sonstige Betriebssystemaufrufe

- Datum / Uhrzeit lesen oder setzen
- Daten zum Netzwerk senden /vom Netzwerk empfangen
- Nutzerrechte abfragen und verwalten

Zugriff auf das Betriebssystem



Zugriff auf das Betriebssystem

- Die Schnittstelle über die auf das Betriebssystem zugegriffen wird, nennt man Application Programming Interface (API).
- Dieses API kann von Anwenderprogrammen genutzt werden.
- Um dem Nutzer des Rechners gewisse Funktionen des Betriebssystems zugänglich zu machen, wird in den meisten Fällen eine sog. Shell mit dem Betriebssystem geliefert. Hierbei handelt es sich um ein kommando- zeilenorientiertes System um Befehle an das Betriebssystem zu geben.
- Neuere Betriebssysteme enthalten statt der kommandozeilenorientierten Befehlseingabe eine grafische Benutzeroberfläche zur Kommunikation mit dem Betriebssystem.

Beispiel für Zugriff auf Betriebssystem (1)

R | R | Z | N

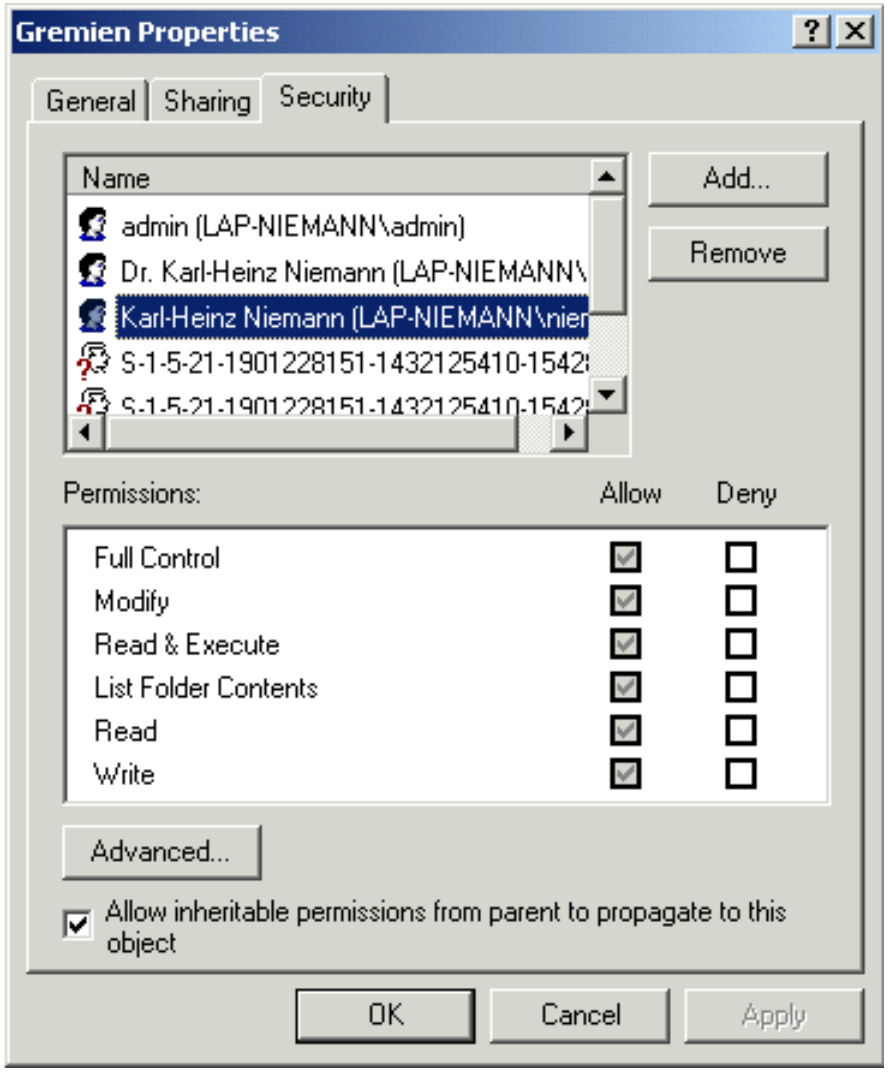
```
C:\>rmdir test

C:\>dir
 Volume in drive C is WIN2000
 Volume Serial Number is 18E9-69BE

 Directory of C:\

16.10.2002  09:25                 0 AdobeWeb.log
29.11.2002  08:22        <DIR>          Benutzervorlagen
03.08.2000  13:11               230 comreads.dbg
03.08.2000  13:11               216 comused.dbg
05.09.2002  10:58        <DIR>          dell
09.10.2002  06:05        <DIR>          Documents and Sett
16.09.2002  08:55        <DIR>          notes
05.11.2002  20:20        <DIR>          Program Files
07.10.2002  10:57                 1 Program1
11.11.2002  12:14        <DIR>          Program_Sources
30.09.2002  13:08                83 ST_F2K_SC.bat
28.11.2002  21:20        <DIR>          temp
17.08.2002  16:18        <DIR>          transfer
24.11.2002  16:56        <DIR>          WINNT
               5 File(s)            530 bytes
               9 Dir(s)   5.471.230.464 bytes free

C:\>
```

Beispiel für Betriebssystemzugriff über kommandozeilen-orientiertes System

Der Befehl **rmdir** löscht Verzeichnisse.

Der Befehl **dir** zeigt den Inhalt des aktuellen Verzeichnisses an.

Beispiel für Zugriff auf Betriebssystem (2)

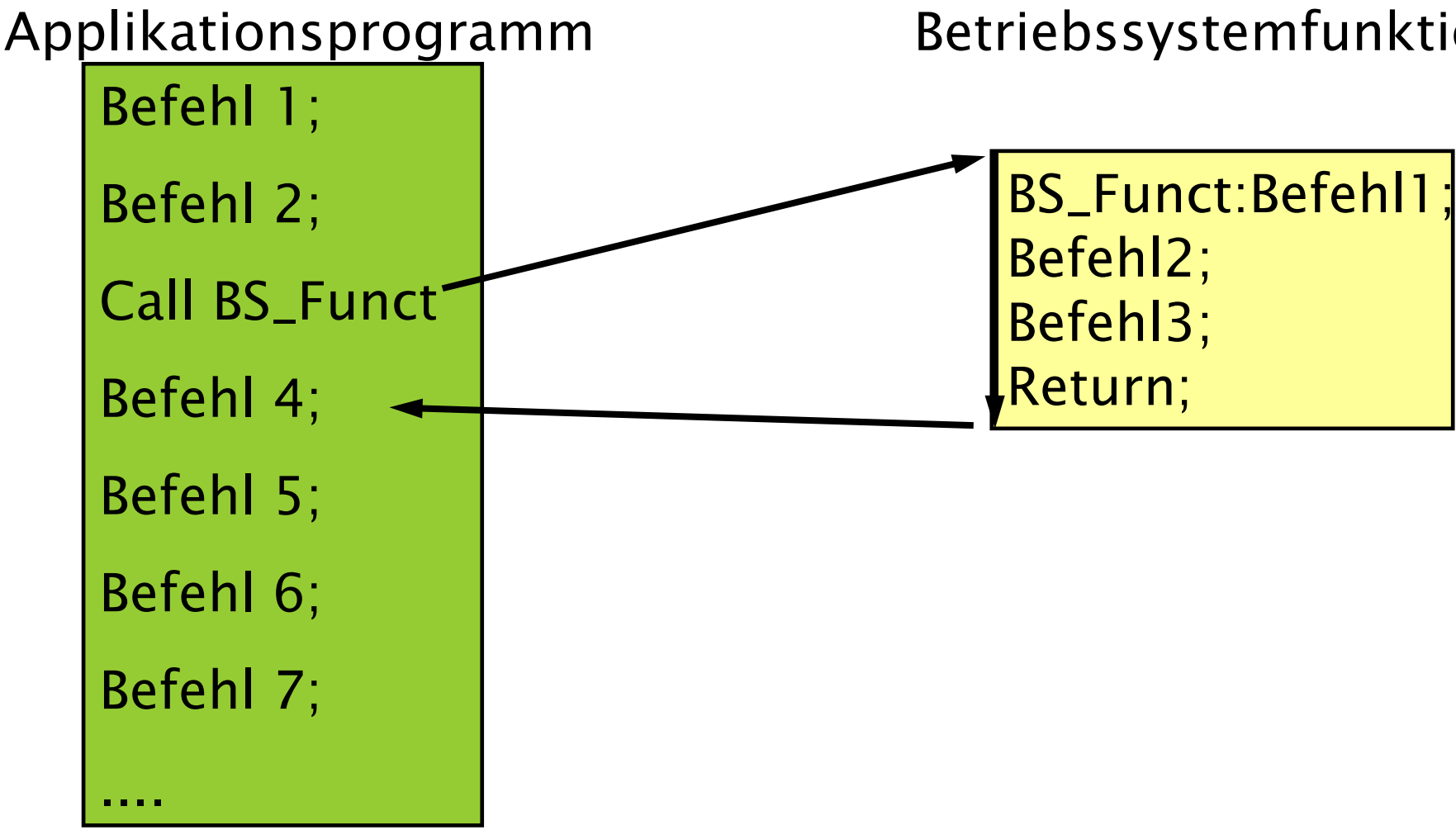


Verwaltung von Zugriffsrechten auf ein Verzeichnis über das grafische User Interface (GUI) in Windows 2000

Beispiel für Betriebssystemaufrufe in Visual Basic zum Öffnen und schließen einer Datei:

```
.....
Open DateiName For Input As DateiNummer
DateiGroesse = LOF(DateiNummer)
.....
.....
Close DateiNummer
```

Beispiel für Zugriff auf Betriebssystem (4)



Formulierung von Aufgaben für die Bearbeitung im Rechner

- Genau definierte Verarbeitungsvorschrift zur Lösung einer Aufgabe.
- Beschreibung eines Schemas, welches unter Verwendung von endlich vielen Arbeitsschritten ein bestimmtes Problem löst.
- Endliche Folge von Anweisungen, die nacheinander ausgeführt werden. Die Anweisungen können unter bestimmten Bedingungen wiederholt werden.

Beispiele aus dem täglichen Leben:

- Kochrezepte
- Steuerprogramme für technische Geräte (z. B. Waschmaschine)

Beispiel für einen Algorithmus: Bedienung einer Waschmaschine

R | R | Z | N

- Tür der Waschmaschine öffnen.
- Max. 5 kg Wäsche (einer Farbe ☺) einfüllen.
- Tür der Waschmaschine schließen.
- Waschmittel passend zur Farbe der Wäsche in die kleine Schublade für den Hauptwaschgang füllen.
- Wasserzulauf öffnen.
- Waschprogramm wählen.
- Starttaste drücken.
- Waschvorgang abwarten.
- Nach Programm-Ende Maschine abstellen.
- Wasserzulauf schließen.
- Tür öffnen und Wäsche entnehmen.

Eigenschaften eines Algorithmus

- Ein Algorithmus benötigt endlich viele Arbeitsschritte.
- Ein Algorithmus ist beschreibbar.
- Jeder Arbeitsschritt ist ausführbar.
- Ein Algorithmus liefert unter identischen Startbedingungen immer das gleiche Endergebnis.
- Der Ablauf des Verfahrens ist eindeutig definiert.

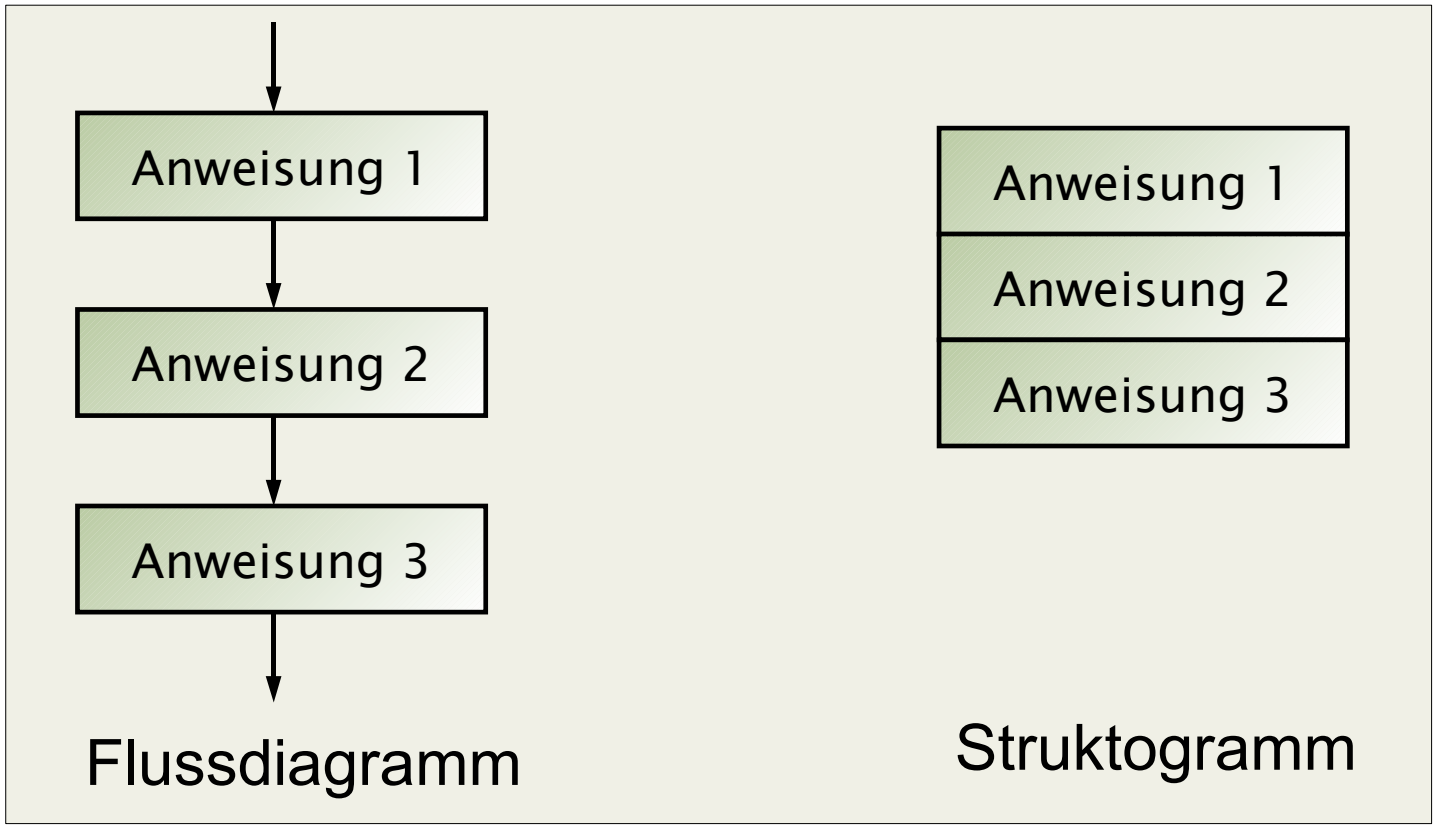
Beispiel: Euklidischer Algorithmus

- Der größte gemeinsame Teiler (ggT) zweier positiver Zahlen wird bestimmt.
- Eingangsdaten: zahl1, zahl2
- Ausgangsdaten: ggT
- Algorithmus:
  1. Eingabe von zahl1 und Speicherung in der Variablen x.
  2. Eingabe von zahl2 und Speicherung in der Variablen y.
  3. Vergleich von x und y:
     - wenn y < x; setze x = x - y und gehe zu 3.
     - wenn x < y; setze y = y - x und gehe zu 3.
     - wenn x gleich y; setze ggT = x
  4. Ausgabe von ggT

- Durch die Ablaufsteuerung wird die Abarbeitung der einzelnen Aktionen festgelegt.
- Abläufe können mit Hilfe von *Flussdiagrammen* oder *Struktogrammen* dargestellt werden.
- Folgende Elemente sind in Flussdiagrammen und Struktogrammen vorhanden:
  - Folge (Sequenz)
  - Auswahl (Selektion)
  - Schleifen (Iteration)
  - Sprünge, um Schleifen vorzeitig zu verlassen oder um zu anderen Algorithmen zu springen.
  - Unterprogrammaufrufe
  - Ein-/Ausgabeoperationen, bzw. Handoperationen

- Darstellung in einem Flussdiagramm .
  - Ein Flussdiagramm ist eine Ablaufdiagramm für Computerprogramme.
  - Die benutzten Symbole sind in der DIN 66001 genormt.
- Darstellung als Struktogramm.
  - Struktogramme werden auch als Nassi-Shneidermann-Diagramme bezeichnet.
  - Die benutzten Symbole sind in der DIN 66261 genormt.
  - Struktogramm-Editoren (Freeware) sind unter http://de.wikipedia.org/wiki/Nassi-Shneiderman-Diagramm zu finden. Der dort aufgeführte HUS-Struktogrammer zeichnet sich durch eine einfache Bedienung aus.

- Eine bestimmte Anzahl von Aktion werden nacheinander ausgeführt.
- Die Anweisungen werden ohne Einschränkung ausgeführt.

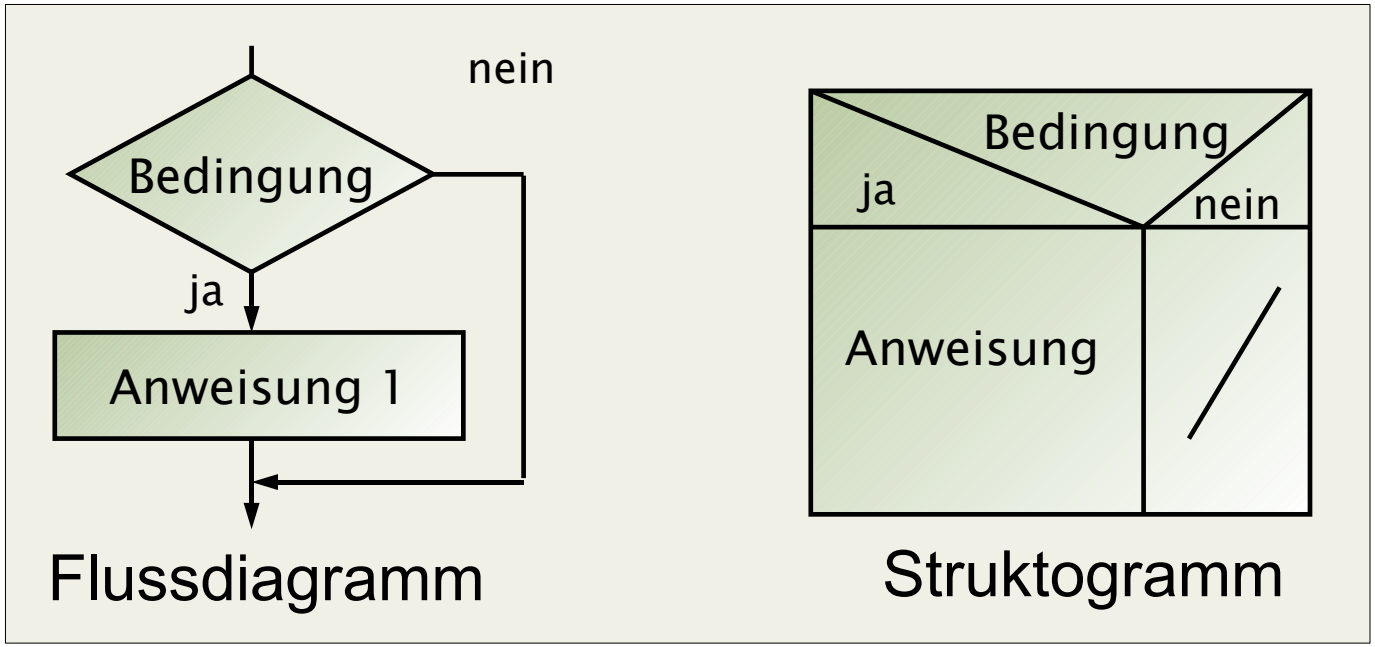


Auswahl (Selektion)

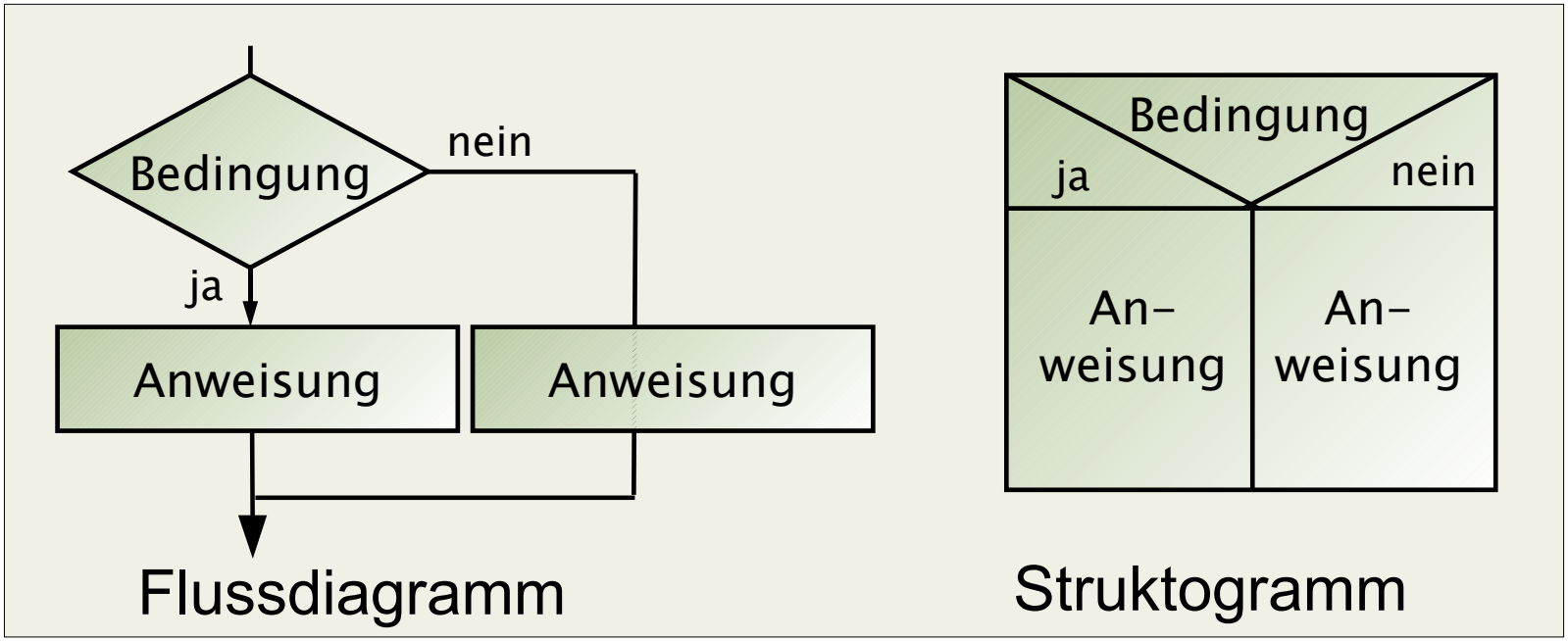
R|R|Z|N

- Die Anweisungen werden nur unter bestimmten Bedingungen ausgeführt.
- Die Aktionen werden nur mit Einschränkungen ausgeführt.
- Man unterscheidet zwischen:
  - Einfache bedingte Anweisung.
  - Vollständige bedingte Anweisung.
  - Fallunterscheidung.

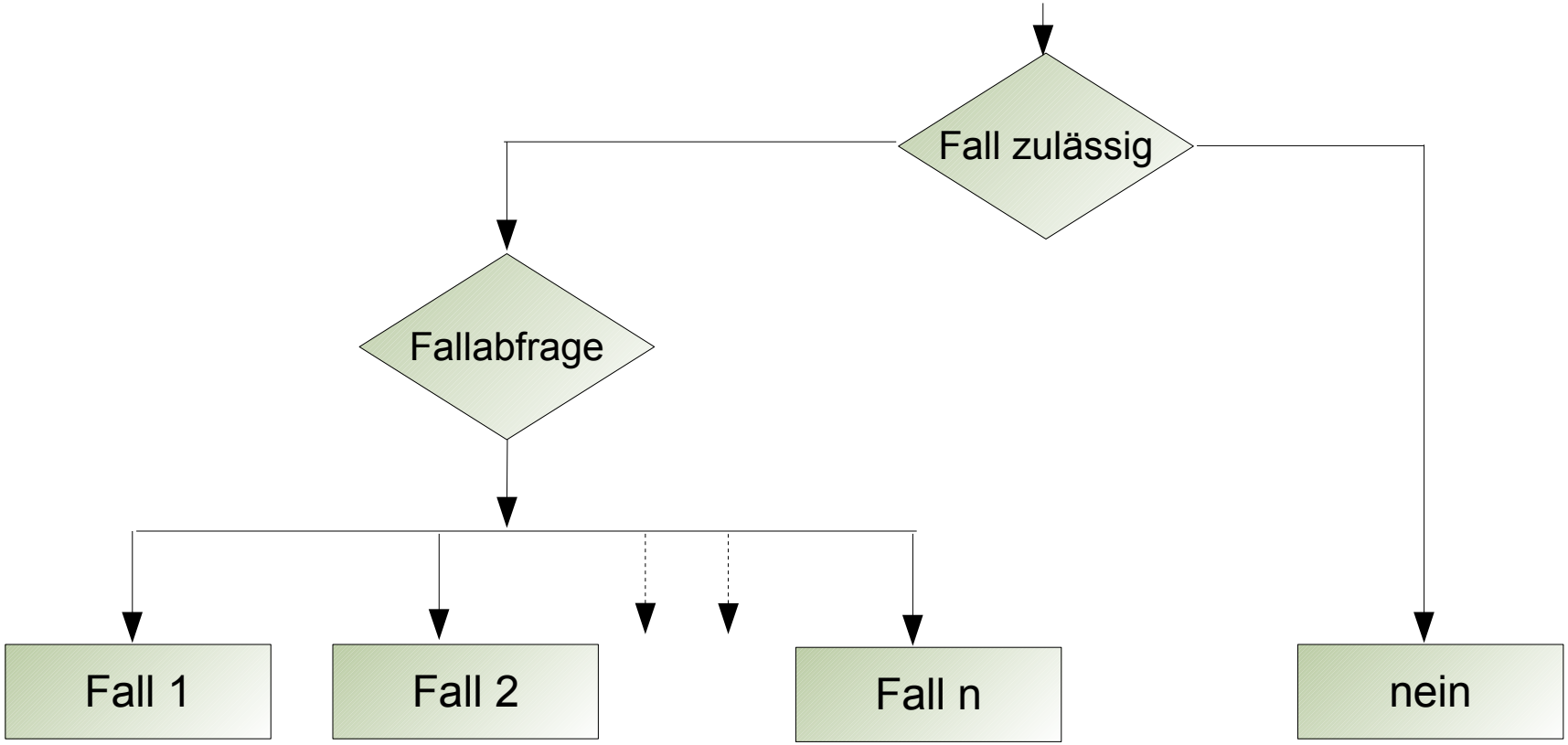
- Die Anweisungen werden nur ausgeführt, wenn die angegebene Bedingung erfüllt ist.



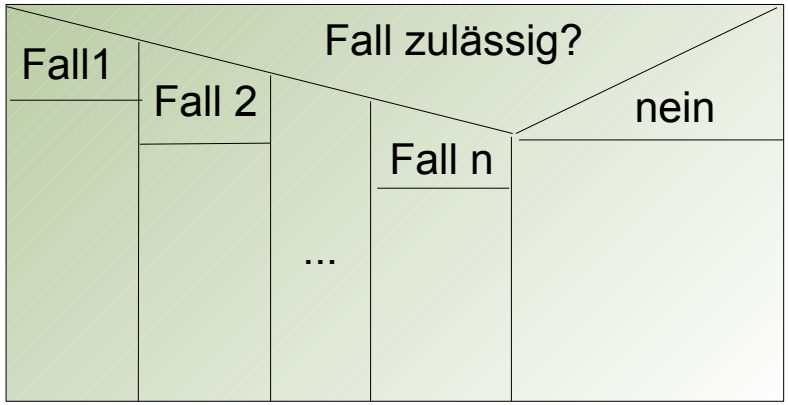
- Es sind Anweisungen vorhanden, die ausgeführt werden, wenn die Bedingung erfüllt ist und andere, wenn die Bedingung nicht erfüllt ist.



- Es sind mehr als zwei Fälle vorhanden. Abhängig von einem Wert werden verschiedene Anweisungen durchgeführt.



- Es sind mehr als zwei Fälle vorhanden. Abhängig von einem Wert werden verschiedene Anweisungen durchgeführt.



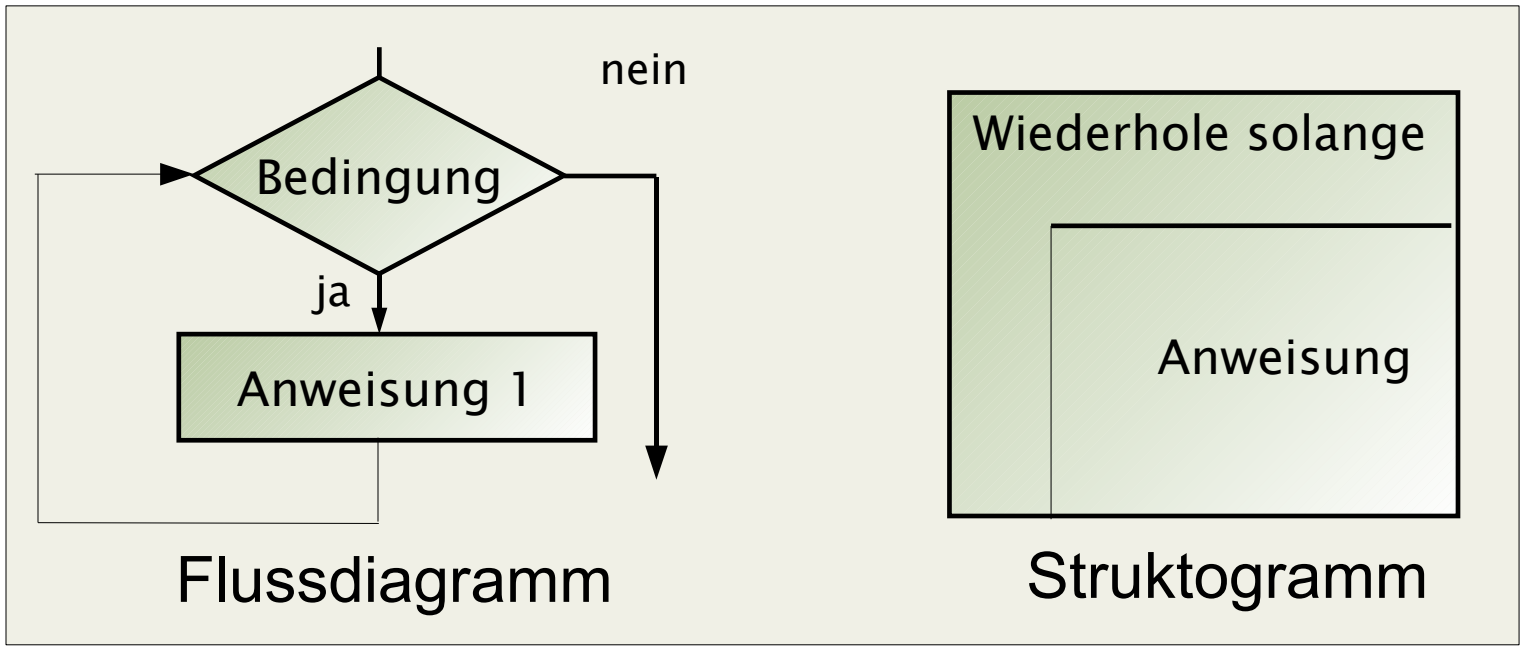
Schleifen (Iteration)

- Schleifen sind eine oder mehrere Aktionen, die in Abhängigkeit von einer Bedingung wiederholt ausgeführt werden.
- Die Schleife wird so lange durchlaufen, bis eine Abbruchbedingung erfüllt wird. Diese wird, je nach Schleifentyp, am Anfang oder am Ende der Schleife geprüft.
- Die Schleife selber besteht aus
  - Anweisungen, die wiederholt ausgeführt werden sollen.

    Eine dieser Anweisungen verändert auch den Wert von Variablen welche die Abbruchbedingung beeinflussen
    (z. B. Stand eines Zählers)
  - Der Entscheidung für den Abbruch oder das Fortsetzen der Schleife anhand der Abbruchbedingung (z. B. des Zählerstandes)

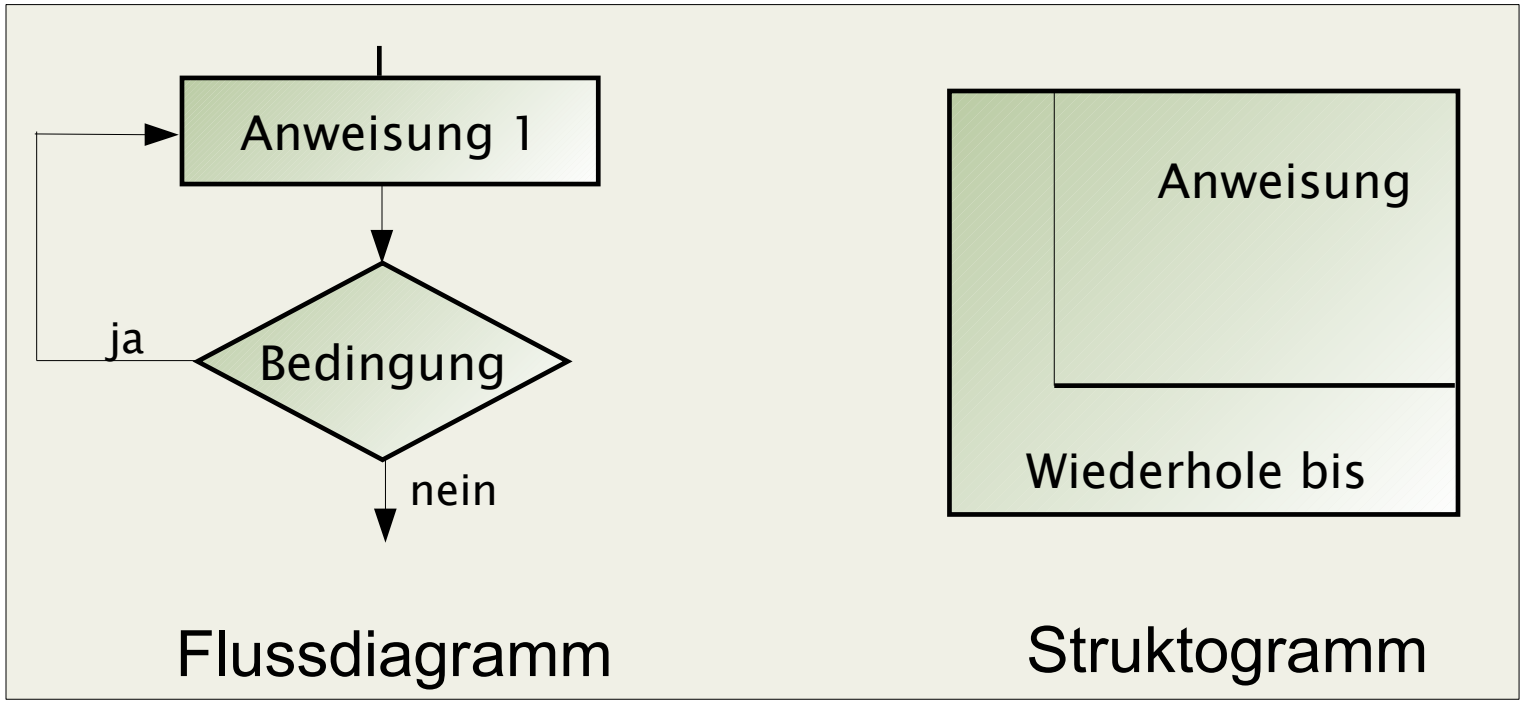
Möglichkeiten für die Definition von Schleifen :

- Solange die Bedingung erfüllt ist, führe die Aktion aus.



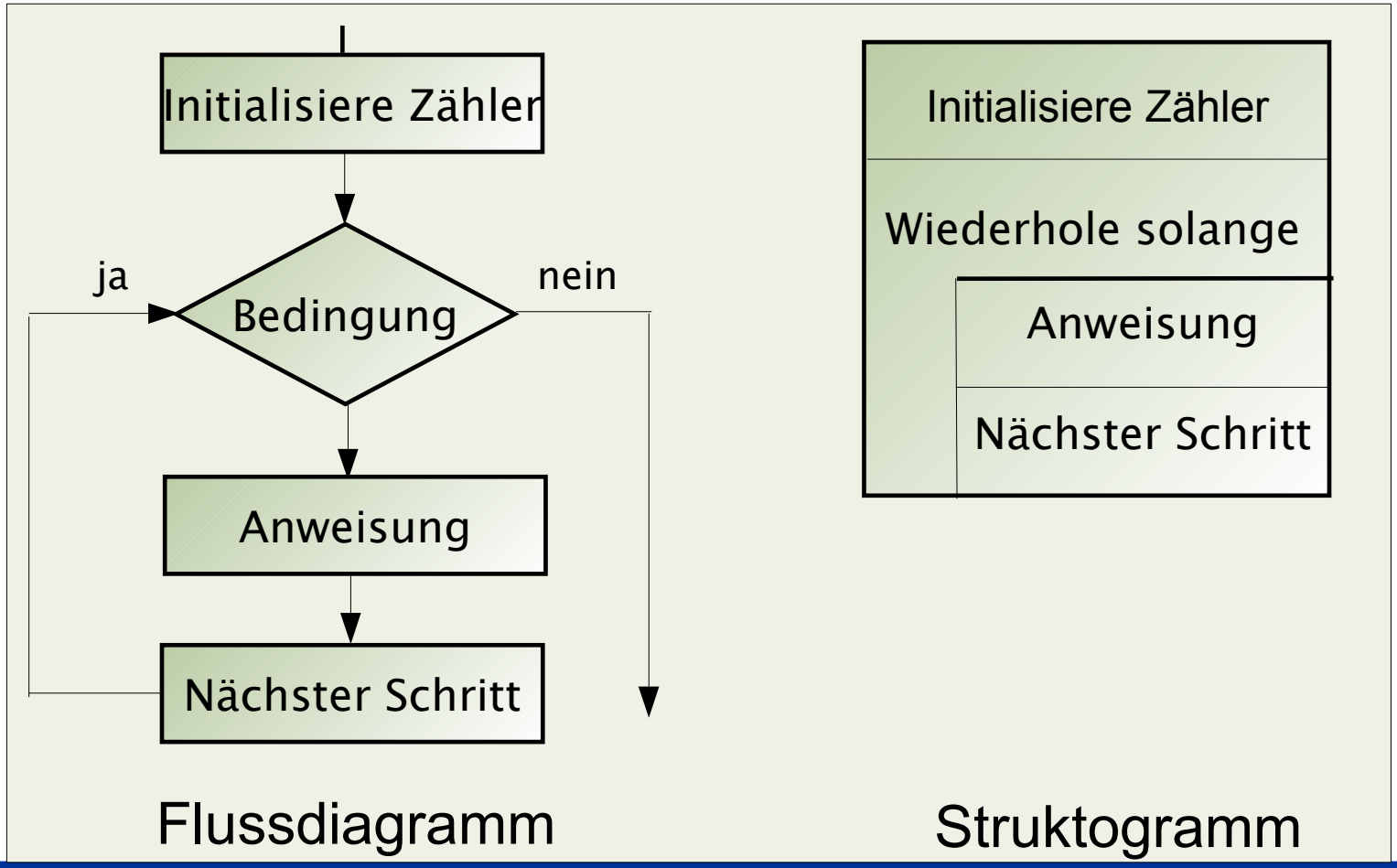
Möglichkeiten für die Definition von Schleifen :

- Bis die Bedingung erfüllt ist, führe die Aktion aus.



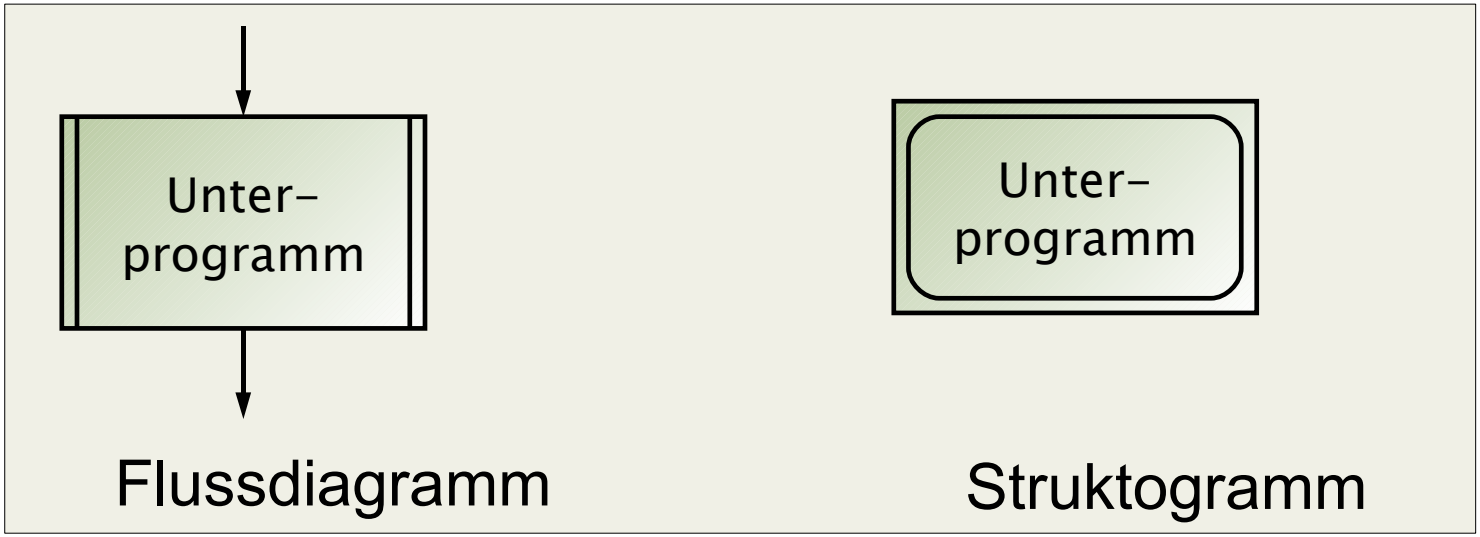
Möglichkeiten für die Definition von Schleifen :

- Zähle von Min bis Max.



Unterprogrammaufruf

- Um die Wartbarkeit von Programmen zu verbessern, sind Aufgaben in Teilaufgaben zu zerlegen. Diese Teilaufgaben können in Form von wiederverwendbaren Unterprogrammen formuliert werden.
- Unterprogramme können vom Aufrufer Werte übergeben bekommen.
- Wenn ein Unterprogramm einen Wert an den Aufrufer zurückgeben, werden sie als Funktionen bezeichnet.
- Wenn ein Unterprogramm keinen Wert an den Aufrufer zurückgeben, werden sie als Prozeduren bezeichnet.



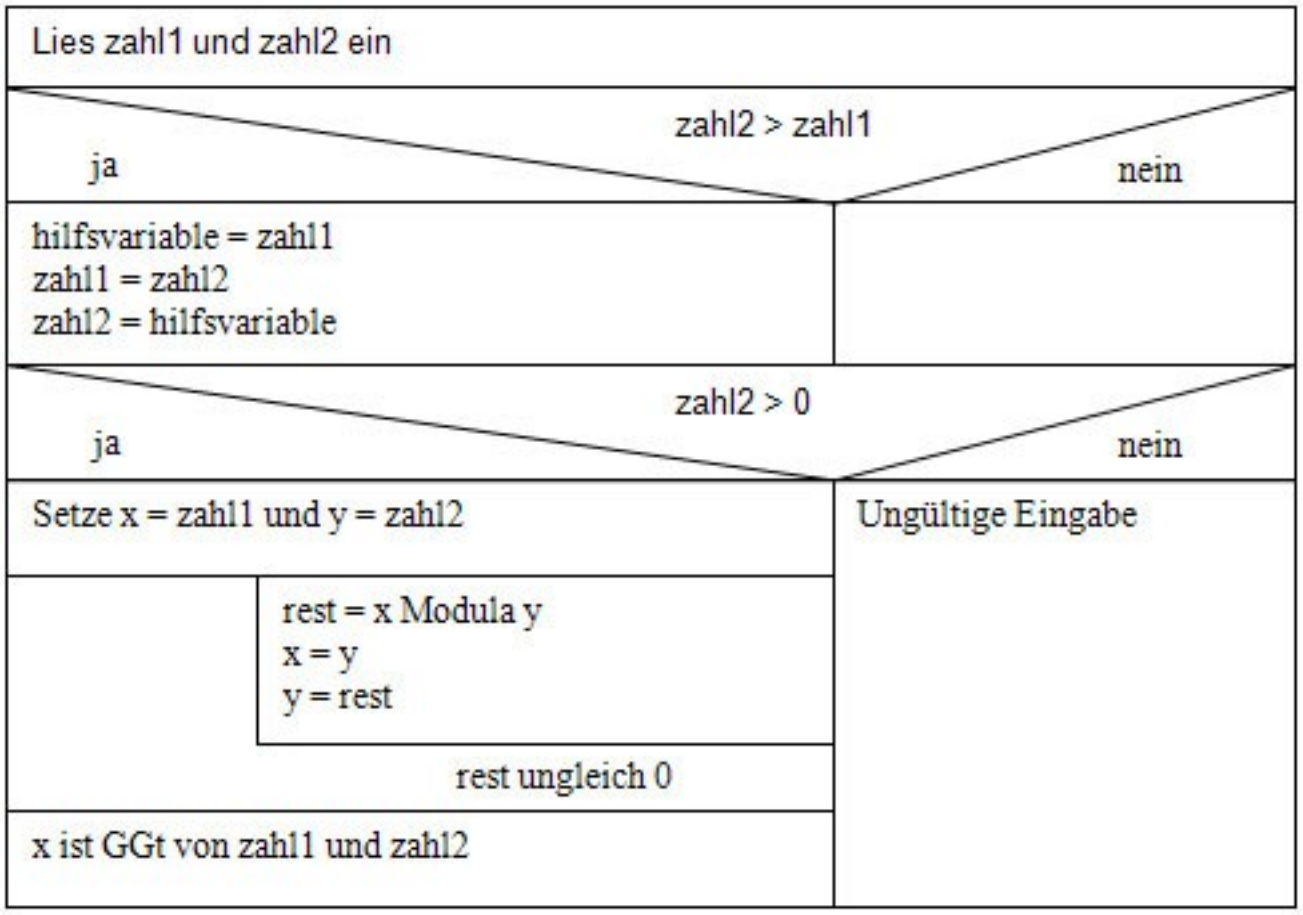
Beispiel: ggT von 38 und 15

- 38 : 15 = 2 Rest 8
- 15 : 8 = 1 Rest 7
- 8 : 7 = 1 Rest 1
- 7 : 1 = 7 Rest 0 ggT = 1

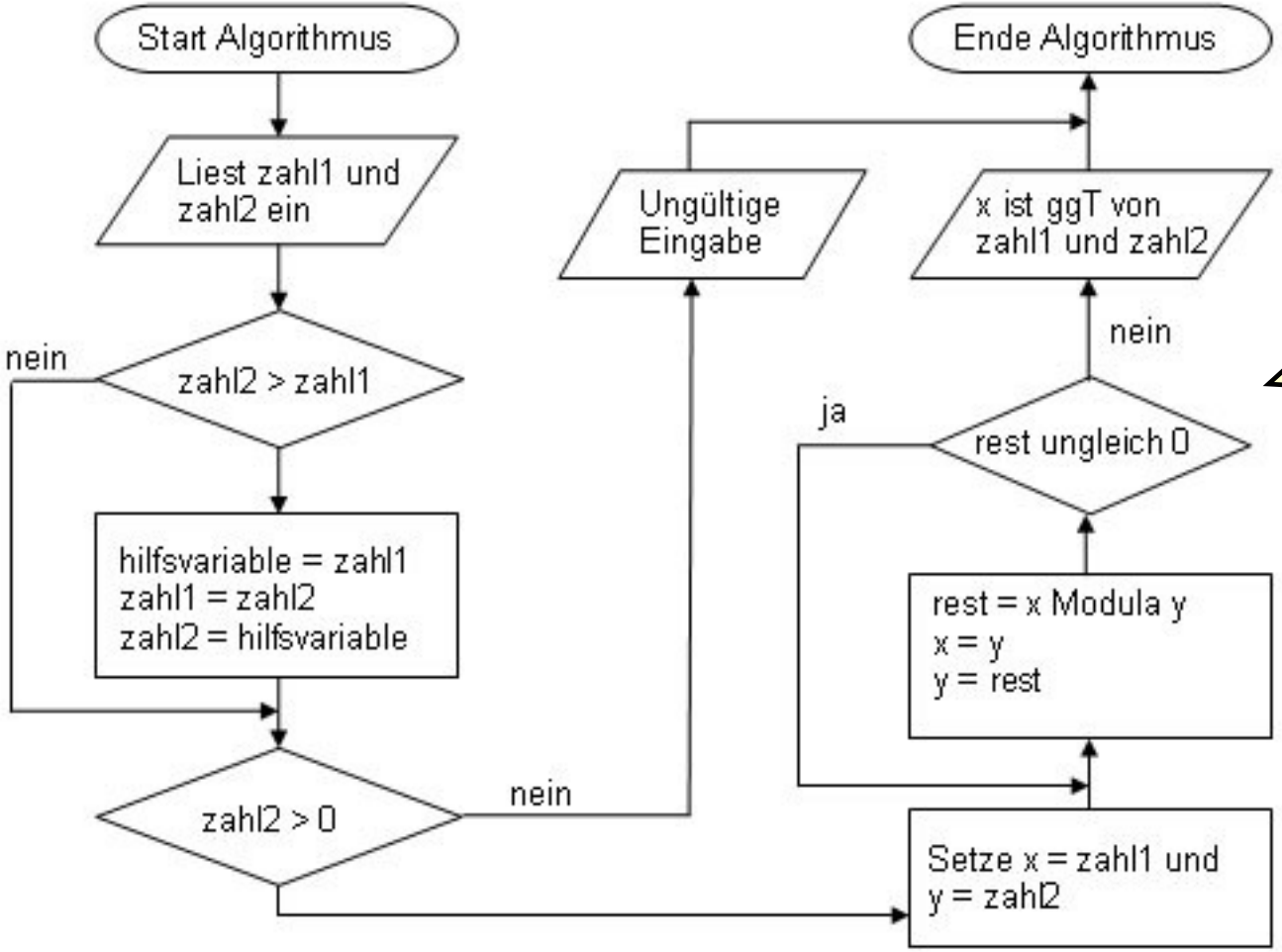
Algorithmus

- als Struktogramm (siehe nächste Folie)
- als Flussdiagramm (siehe übernächste Folie)

Euklidischer Algorithmus (Struktogramm)

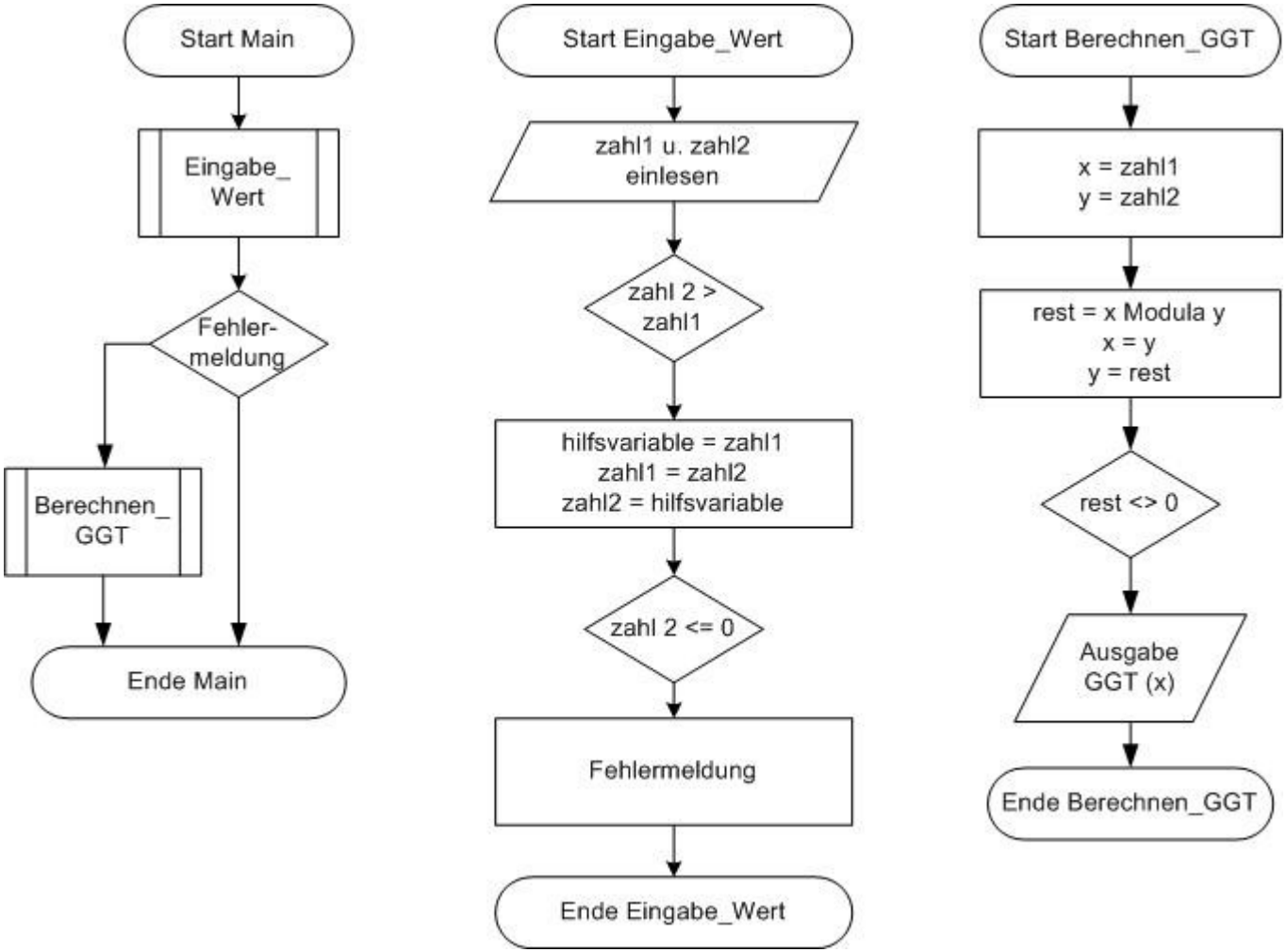


Euklidischer Algorithmus (Flussdiagramm)



Wie könnte man dieses Diagramm besser zeichnen?

Euklidischer Algorithmus (Flussdiagramm mit Unterprogrammen)



Vorteile von Struktogrammen

- Struktogramme erzwingen einen sequentiellen Programmablauf ohne Sprünge.
- Im Allgemeinen führt die Verwendung von Struktogrammen dazu, dass Programme besser strukturiert sind, als bei der Verwendung von Programmablaufplänen.
- Verwenden Sie vorzugsweise (zumindest während Sie ihre erste Programmiersprache lernen) Struktogramme.
- Machen Sie erst einen Programmentwurf bevor Sie mit der Codierung beginnen.

Programme und deren Sprache

R|R|Z|N

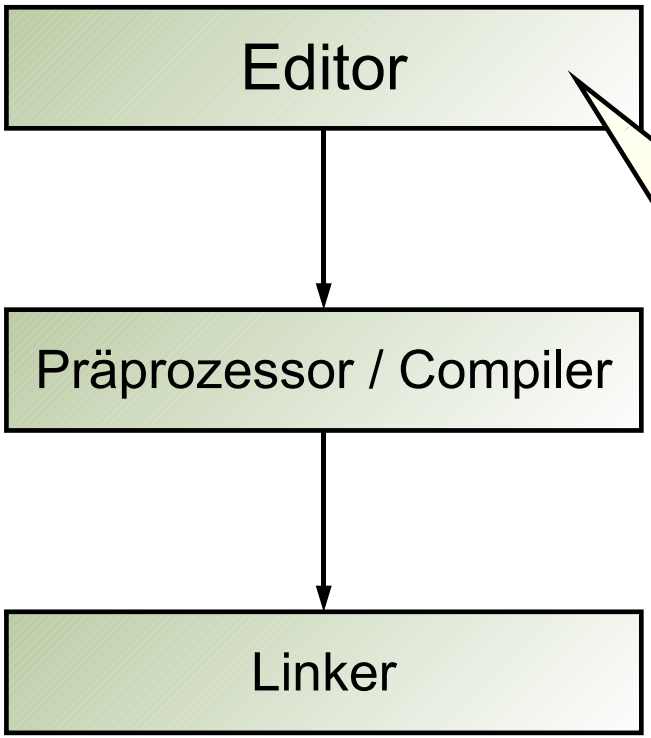
- Ein Computerprogramm
  - ... setzt einen Algorithmus in eine Form um, die von einem Computer verarbeitet werden kann.
  - ... teilt dem Computer mit, was er und wie er eine Aufgabe zu lösen hat.
- Der Begriff Programm
  - ... wird für einen Text genutzt, der in einer Programmiersprache abgefasst ist. Dieser Text wird als Quelltext bezeichnet und kann von einem Menschen gelesen werden.
  - ... wird für den von einem Computer ausführbaren Maschinencode verwendet. Um aus dem Quelltext einen Maschinencode zu genieren, muss dieser übersetzt werden.
- In diesem Kurs werden Computerprogramme in der Programmiersprache C geschrieben.

- 1971 entwickelten Brian W. Kernighan und Dennis Ritchie in den Bell-Labororien (Murray Hill, USA) die Programmiersprache C auf einem PDP-11-Rechnersystem.
- In den Jahren 1983 bis 1989 wurde diese Sprache, die heute im allgemeinen Sprachgebrauch als ANSI C oder C89 bezeichnet wird, vom American National Standard Institute (ANSI) erweitert und als Norm verabschiedet. ANSI C hat einen Sprachumfang von 32 Schlüsselwörtern und ist als Teilmenge in C++ enthalten.
- Zur Laufzeit eines C-Programms findet keine automatische Fehlerüberprüfung statt.
- ANSI C ist eine strukturierte Sprache, die eine Aufgabe in Teilaufgaben zerlegen kann. Die Teilaufgaben werden vor dem Rest verborgen.

- Im Jahr 1990 wurde dieser Standard mit kleinen Änderungen von der International Standards Organization (ISO) übernommen.
- Im Jahr 1995 wird die C-Standardbibliothek um eine Sammlung von Funktionen zur Programmierung von Zeichen, die durch 16- und 32-Bit-Zeichencodes codiert werden, ergänzt. Die meisten dieser Funktionen werden in den Header-Dateien wchar.h und wctype.h definiert.
- Im Jahr 1999 wird ein neuer Standard verabschiedet, der als C99 bezeichnet wird. Mit diesem Standard wurden neue Schlüsselwörter, Zeiger, Datentypen und Bibliotheken eingeführt.
- Die meisten Programme (Compiler etc.) nutzen noch heute den C89-Standard.

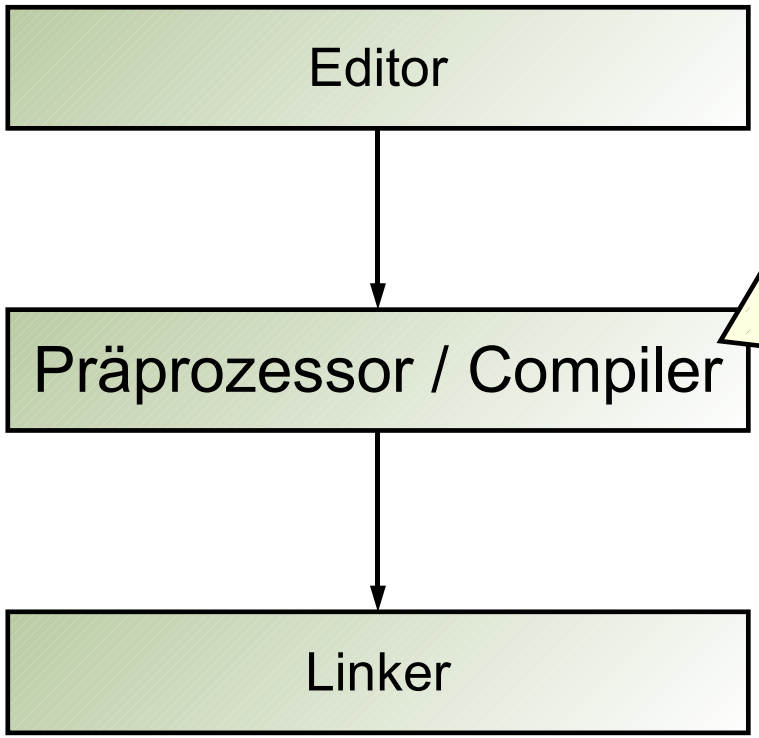
- C-Quellprogramme können sehr maschinennah geschrieben werden.
- C-Quellprogramme nach dem ANSI-Standard sind weitgehend portabel, d. h. auf verschiedenen Betriebssystemen und CPU-Typen lauffähig.
- Ein C-Programm kann einen Algorithmus in verschiedene Teile zerlegen, die sich untereinander aufrufen und gemeinsam ein Problem lösen.
- Viele Fehler werden vom Übersetzer eines C-Programms nicht erkannt. Diese können zur Laufzeit des Programms zu Problemen führen.
- C-Programme können sehr kompakt geschrieben werden. Die Lesbarkeit des Quelltextes verschlechtert sich aber dadurch.

Vom C-Programm zum Maschinencode



- Der Quelltext kann mit Hilfe eines beliebigen Texteditors geschrieben werden.
- Der Text wird im Format "Text" gespeichert.
- Die Datei hat die Endung ".c".
- Häufig werden heute Entwicklungsumgebungen für das Entwickeln von C-Programmen genutzt, die z. B. über Funktionen wie „Auto-Vervollständigen“ und Syntaxkennzeichnung verfügen.

Vom C-Programm zum Maschinencode



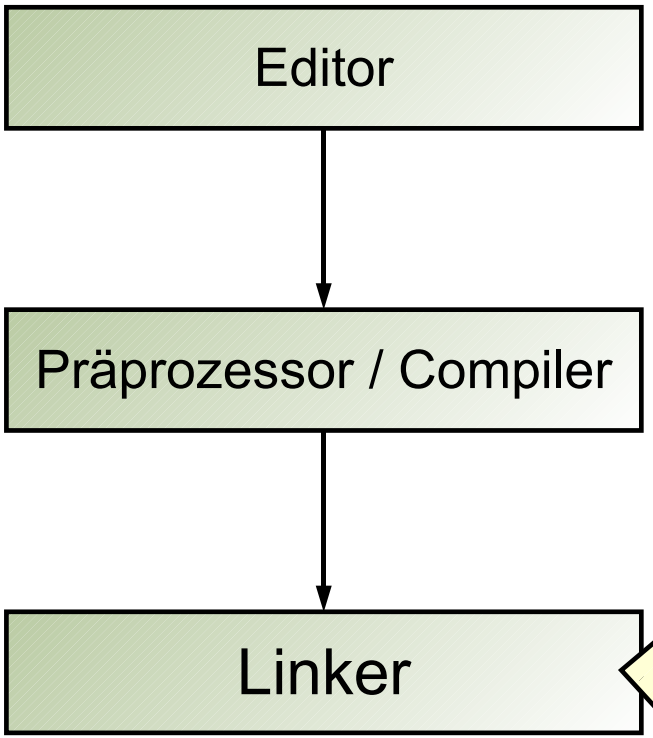
Präprozessor:

- Bestandteil des Compilers.
- Verarbeitung von Zeilen, die mit einem # beginnen.
- Suchen und Ersetzen-Funktionen von Links zu Dateien und symbolischen Variablen.
- Bevor der Compiler seine Arbeit aufnimmt, wird Text eingefügt und ersetzt. aufnimmt.

Compiler:

- Übersetzungsprogramm.
- Programme, die in einer für den Menschen verständlichen Sprache geschrieben sind, werden in Maschinensprache übersetzt.
- Diese Maschinensprache enthält Befehle, die direkt vom Prozessor ausführbar sind.
- Der Computer führt nur diese Maschinenbefehle aus.

Vom C-Programm zum Maschinencode



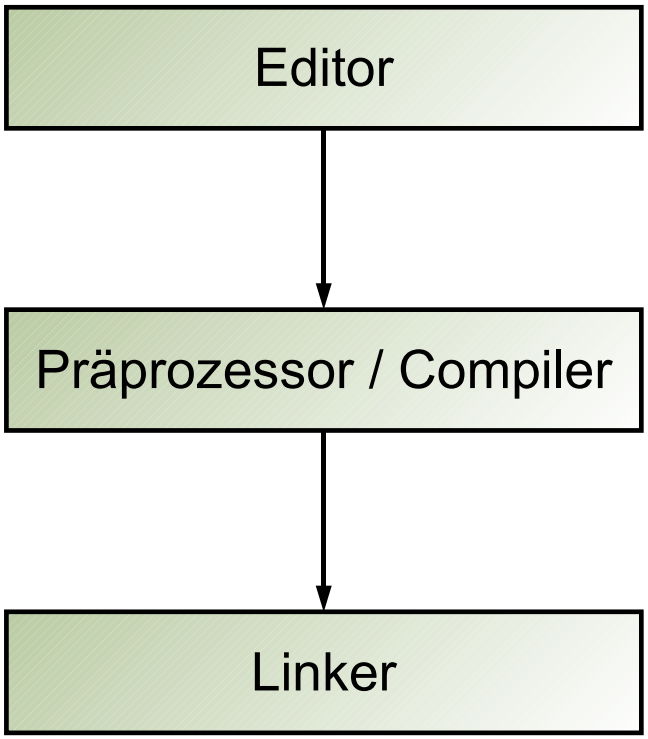
Linker:

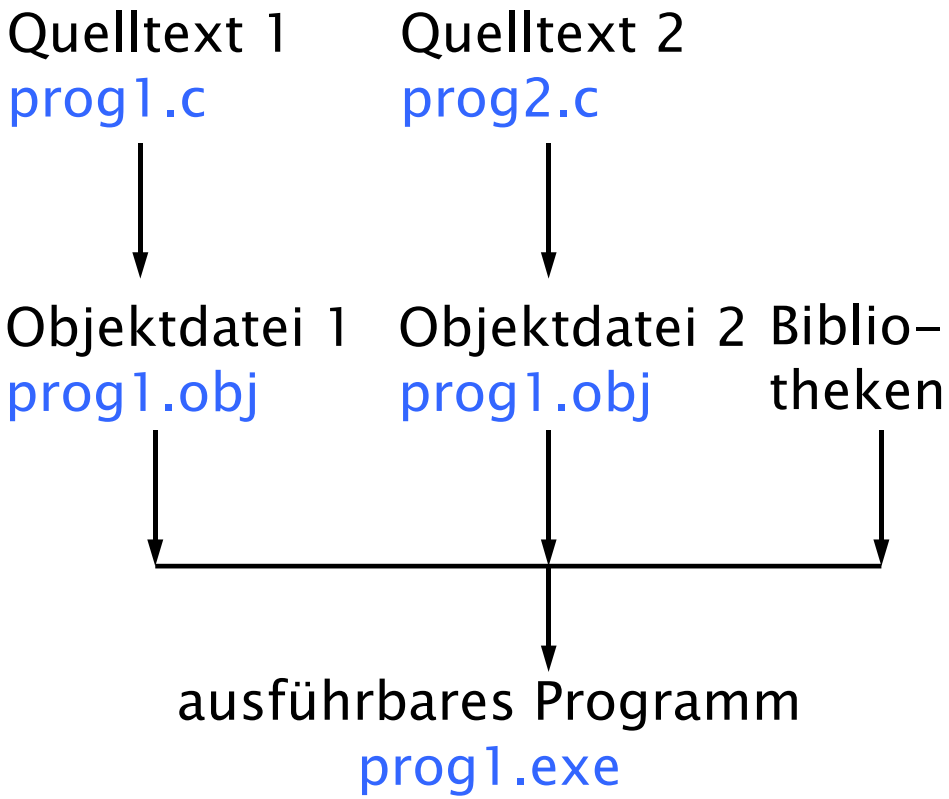
- Zubehörprogramm zu einem Compiler.
- Erstellt aus dem Objektcode und Bibliotheken ein lauffähiges Programm.
- Aus verschiedenen Modulen wird ein lauffähiges Programm erzeugt (gebunden).

Bibliotheken:

- Funktionen für bestimmte Aufgaben werden in einer Datei zusammengefasst.
- Die Funktionen kann ein Programmierer in sein eigenes Programm einbinden.
- In einer Bibliothek für die Ein- und Ausgabe von Daten, findet man z. B. vordefinierte Funktionen zur Ein- oder Ausgabe von Text auf den Bildschirm.

Vom C-Programm zum Maschinencode





Vom C-Programm zum Maschinencode

```
4012eb    movb   $0x2,-8(%ebp)
[    7]  i=0xFFFF;
4012ef    movb   $0x5,-8(%ebp)
4012f3    movl   $0xffff,-16(%ebp) (i)
[    8]  j=0x1234;
4012fa    incl   -8(%ebp)
4012fd    movl   $0x1234,-20(%ebp) (j)
[    9]  k=0x5678;
[   10]
401304    incl   -8(%ebp)                          ff45f8
401307    movl   $0x5678,-24(%ebp) (k)             c745e87856000
[   11]  for (i=1;i<10;i++)
[   12]          {
40130e    movb   $0x9,-8(%ebp)                    c645f809
401312    movl   $0x1,-16(%ebp) (i)                c745f0010000
[   13]          j=j+1;
401319    movb   $0xb,-8(%ebp)                    c645f80b
40131d    incl   -20(%ebp) (j)                     ff45ec
[   14]          k=k+1;
401320    incl   -8(%ebp)                          ff45f8
401323    incl   -24(%ebp) (k)                     ff45e8
[   11]  for (i=1;i<10;i++)
[   12]          {
[   13]          j=j+1;
[   14]          k=k+1;
[   15]          }
401326    movb   $0x9,-8(%ebp)                    c645f809
40132a    incl   -16(%ebp) (i)                     ff45f0
40132d    cmpl   $0xa,-16(%ebp) (i)                837df00a
401331    jl     401319                            7ce6
[   16]  return 0;
401333    movb   $0xe,-8(%ebp)                    c645f80e
401337    movl   $0x0,%eax                         b800000000
```

Aus dem C-Code:
for (i=1;i<10;i++)
{j=j+1; k=k+1;}

wird dieser Asembler Code generiert

Und das bekommt der Mikro-prozessor davon zu sehen

Quelldateien

- ... haben die Dateiendung ".c".
- ... enthalten das in C geschriebene Programm.
- ... sind für Menschen, die die Sprache verstehen lesbar.

```
#include <stdio.h>
main()
{
   int variable;

   variable = 1;
   printf ( "1 = %d\n", variable );
}
```

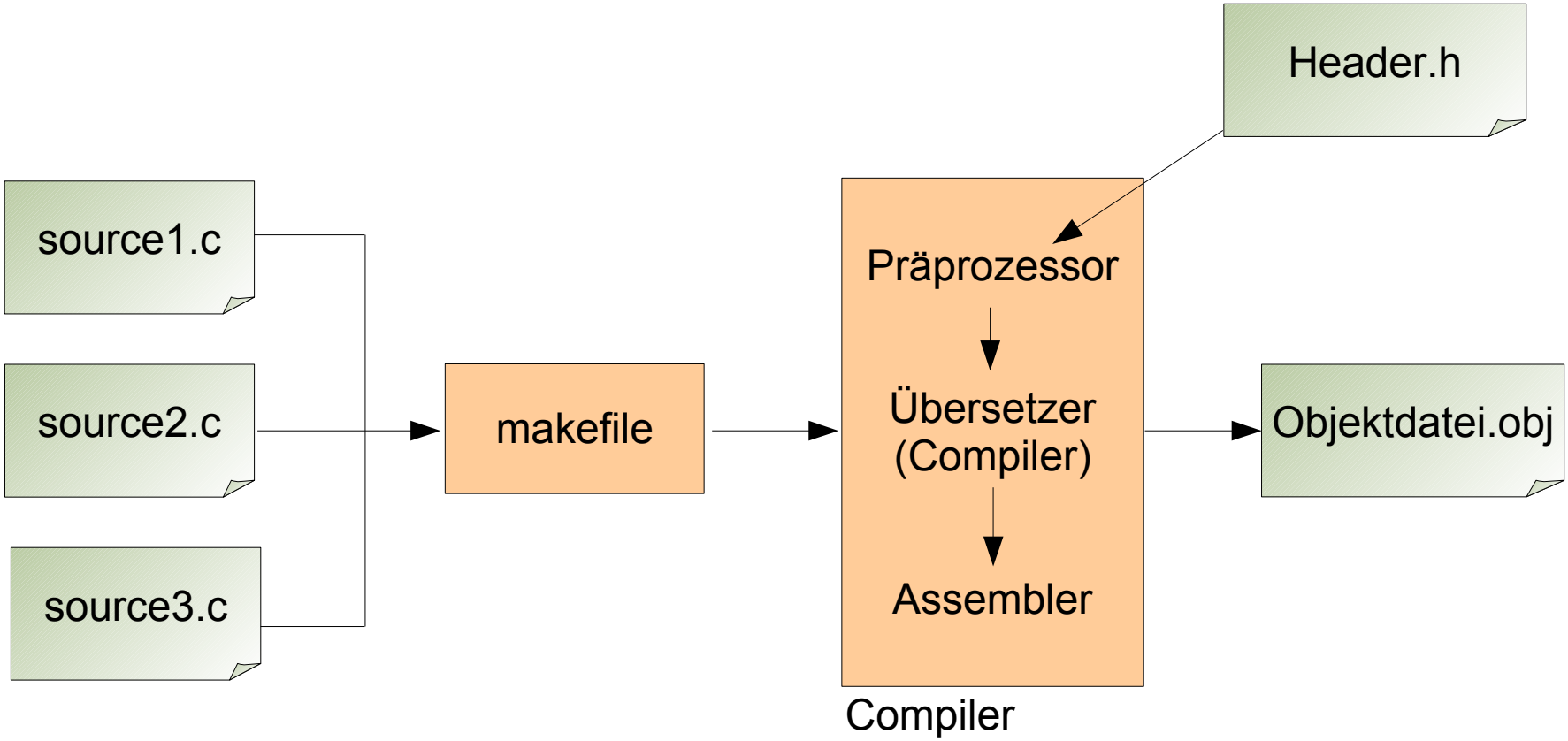
- ... haben die Dateiendung ".h".
- ... enthalten Deklarationen von Variablen und Funktionen, die in vielen verschiedenen Dateien benötigt werden.
- ... enthalten in C geschriebenen Code, der nicht vom Compiler übersetzt werden muss.

```
#ifndef _STDIO_H_
#define  _STDIO_H_

/* All the headers include this
file. */
#include <_mingw.h>

#ifndef RC_INVOKED
#define __need_size_t
#define __need_NULL
#define __need_wchar_t
```

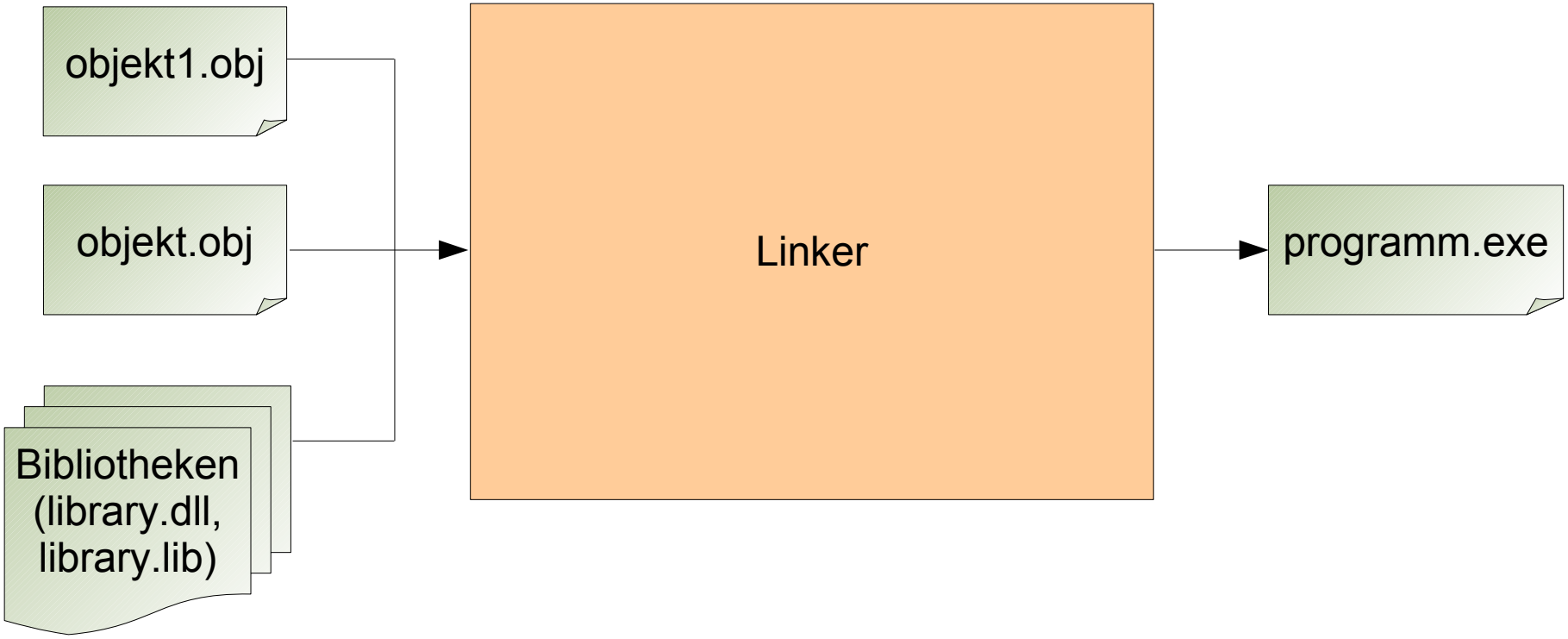
- ... enthält Befehle, um Objektdateien und ausführbare Dateien zu erstellen.
- ... enthält Befehle in einer skript-ähnlichen Sprache, die Dateien, die zu einem Thema gehören, zusammen bindet.

```
all : myMain.o
    gcc -o myMain myMain.o

myMain.o : myMain.c
    gcc -ggdb -c myMain.c

clean :
    -del myMain.o
```

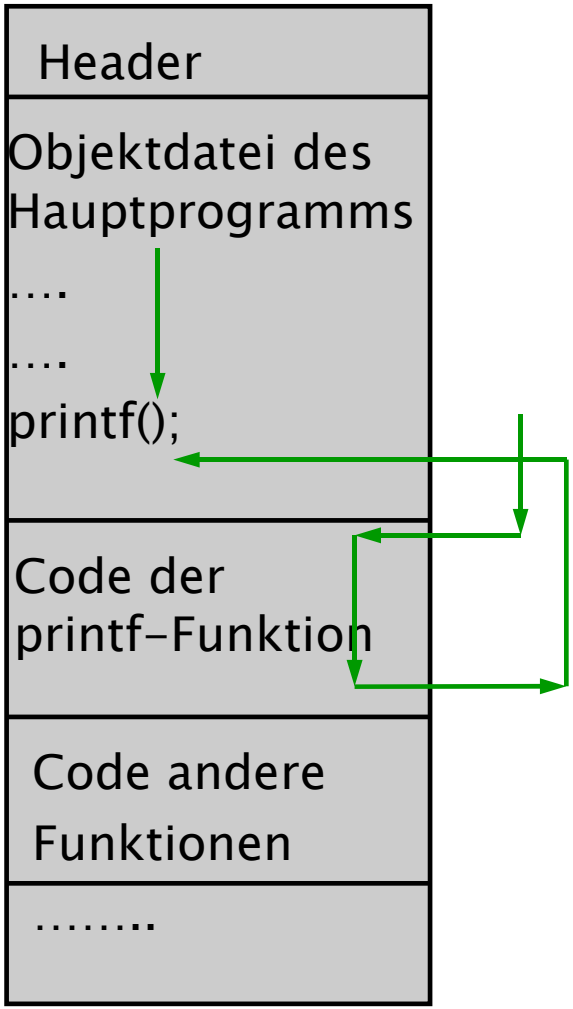
Objektdatei erstellen



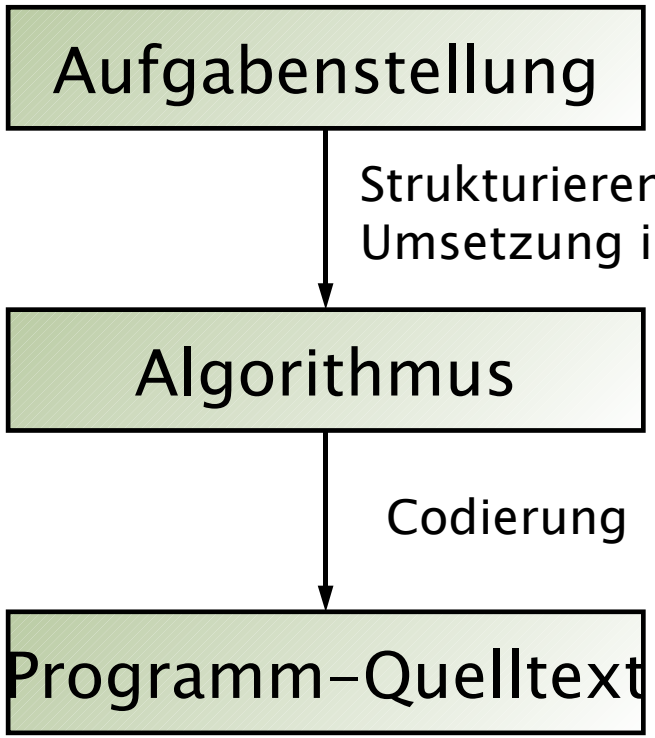
Ausführbare Datei erstellen



Der Linkvorgang



- Beim Linkvorgang werden an den Stellen, wo externe Funktionen aufgerufen werden, die Zieladressen dieser Funktionen eingetragen.
- Dies kann statisch, d. h. während des Linkvorgangs oder dynamisch zur Laufzeit des Programms erfolgen.



Lastenheft, Pflichtenheft,
Funktionale Spezifikation,

Design-Spezifikation,
Test-Spezifikation

Überlegen Sie sich, welches Problem das Programm lösen soll.

- Strukturieren Sie Ihre Aufgabenstellung.
- Welche Funktionen soll das Programm haben?
- Welche Eingangs- und Ausgangsparameter erhält das Programm?
- Zerlegen Sie die Aufgabe in Teilaufgaben.
- Erstellen Sie eine Struktogramm für das Programm.

Überlegen Sie sich eine Teststrategie.

- Wie kann ich das Programm testen?
- Was sind zulässige Eingabewerte, was sind unzulässige Eingabewerte.
- Wie verfährt das Programm mit unzulässigen Eingabewerten?
- Wie kann ich alle Programmzweige im Programm testen.
- Ändern Sie ggf. das Struktogramm, wenn keine Testbarkeit gegeben ist.

Bilden Sie das Struktogramm anschließend in ein Programm ab.

- Der Quellcode wird mit Hilfe eines Texteditors oder einer Entwicklungsumgebung geschrieben. In der Regel haben Quellcode-Dateien in C die Endung ".c" und sind im Format "Text" abgespeichert.

Testen Sie das Programm gemäß Ihrer vorab definierten Teststrategie

- Dokumentieren Sie die Durchführung der Tests und die Ergebnisse.
- Führen Sie Fehlerlisten, der noch zu behebenden Fehler.
- Enthält Ihr Programm Fehler, müssen Sie das Programm mit Hilfe eines Debuggers analysieren, um die Fehler zu finden und zu bereinigen.
- Stellen Sie sicher, dass Sie alle Alternativen im Programmzweig während Ihres Tests durchlaufen.
  - Ein Programm, bei dem nicht alle Pfade vollständig getestet sind, ist nicht getestet.
  - Testen Sie zum Abschluss des Projektes, nach Durchführung aller Korrekturen noch einmal das ganze Programm.

Arbeiten Sie NICHT nach dem CFTL*-Verfahren gemäß der Devise lieber zweimal messen und nur einmal sägen.

***)CFTL: Code first, think later ☺**

Installation eines GNU C-Compilers

R|R|Z|N

- Installieren Sie den GNU C-Compiler.
- Wenn Sie mit Windows XP arbeiten, öffnen Sie die *Systemsteuerung / System*. Wechseln Sie auf die Karte *Erweitert* und klicken dort auf die Schaltfläche Umgebungsvariablen. Der Wert der Variablen PATH wird um den Eintrag *Laufwerk:\Programme\Ordner_des_Compilers\bin* erweitert.

Programm schreiben und kompilieren

R|R|Z|N

- Schreiben Sie das Programm mit Hilfe eines Texteditors und speichern es ab.
- Kompilieren Sie das Programm in der MS-Eingabeaufforderung mit folgenden Befehl

  gcc -o Pfad_ausführbare_Datei Pfad_C-Quelldatei.
- Falls Fehler vorhanden sind, muss das Programm mit folgender Zeile in der Eingabeauffordung kompiliert werden:

  gcc -o -g Pfad_ausführbare_Datei Pfad_C-Quelldatei.

Programm debuggen

- Mit **gdb Pfad_ausführbare Datei** wird der GNU-Debugger gestartet.
- Mit Hilfe von **run** wird das ausführbare Programm gestartet und **quit** beendet es.
- **list** zeigt das Programm an und nummeriert die Zeilen des Programms durch .
- **step** führt das Programm zeilenweise aus.
- Hilfe zu anderen wichtigen GNU-Debugger-Befehlen finden Sie unter http://www.stud.uni-potsdam.de/~hoeffi/gdb.html.

Programm starten

R|R|Z|N

- Führen Sie das Programm aus.
- Sie sollten sich anhand von Tests vergewissern, dass das Programm erwartungsgemäß funktioniert.
- Wenn nicht, beginnen Sie wieder von vorn und nehmen die entsprechenden Änderungen oder Erweiterungen an Ihrem Quelltext vor.

Das erste Programm

R|R|Z|N

```
/*************************************************************
 * Filename: hello.c                                         *
 * Beschreibung: Ausgabe von Hello World                     *
 * History :    Date              Reason                     *
 *               04-06-05         Created                    *
 *************************************************************/

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

Kommentare

```
/*********************************************************
 * Filename: hello.c                                     *
 * Beschreibung: Ausgabe von Hello World                 *
 * History :    Date              Reason                 *
 *               04-06-05         Created                *
 *********************************************************/

#include <stdio.h>

/* Beginn des Pr
```

- Ein Programm oder eine Routine sollte immer mit einem Kommentar beginnen.
- Der Kommentar enthält den Programmnamen, das Datum der Erstellung und / oder letzten Änderung, den Namen des Erstellers / der Erstellerin sowie eine kurze Beschreibung, welches die Funktionalität des Programms beschreibt.
- Kommentare beginnen mit /* und enden mit */.
- Kommentare können mehrzeilig sein.
- Kommentare beeinflussen die Performance des Programms nicht.

Kommentare

```
/*************************************************************
 * Filename: hello.c                                         *
 * Beschreibung: Ausgabe von Hello World                     *
 * History :    Date                Reason                   *
 *               04-06-05           Created                  *
 *************************************************************/

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

- Diese Art der Kommentierung ist aufwendig zu erstellen.
- Beim Einfügen von Worten müssen die Sternchen am rechten Rand neu ausgerichtet werden.

Kommentare

R|R|Z|N

```
/*************************************************************
   Filename: hello.c
   Beschreibung: Ausgabe von Hello World
   History :    Date                Reason
                04-06-05            Created
 *************************************************************/

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

- Dieser Kommentar ist einfacher zu pflegen
- Es lässt sich einfach Text ergänzen, ohne dass Zeichen am rechten Rand editiert werden müssen.

Kommentare

```
/*************************************
   Filename: hello.c
   Beschreibung: Ausgabe von Hello Wor
   History :    Date                Re
                04-06-05            C
 *************************************

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

- Zwischen dem Sternchen und dem Schrägstrich darf es kein Leerzeichen geben.
- Der Compiler ignoriert den gesamten Text zwischen Beginn und Ende des Kommentars.
- Im Kopf des Programms steht ein mehrzeiliger Kommentar. Hier wird eine einzige Kommentarzeile genutzt.
- Kommentare können an beliebiger Stelle eines Programms platziert werden. Sie sollten aber immer die Zeile oder den folgenden Programmabschnitt erläutern.
- Ein Kommentar darf keinen weiteren Kommentar enthalten, d.h. kein Kommentar im Kommentar.

Kommentare

R|R|Z|N

- Kommentare sollen dazu führen, dass man das Programm ohne fremde Hilfe verstehen kann.
- Kommentare sollen nicht offensichtliche Dinge in Prosa wiederholen. Ein negatives Beispiel:
  - a = c + b; /* a ist die Summe von c und b */
- Kommentare müssen bei Änderungen am Programm aktualisiert werden.
- Kommentare sollen sich auf das Warum beziehen und weniger auf das wie.
- Keine redundanten oder überflüssigen Kommentare.

Das erste Programm

```
/*********************************
   Filename: hello.c
   Beschreibung: Ausgabe von He
   History :    Date
                04-06-05
 ************************

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

- Am Anfang eines Programms stehen Präprozessor-Anweisungen. Diese Anweisungen beginnen immer mit einem #.
- Jede Anweisungen steht in ihrer eigenen Zeile und endet nicht mit einem Semikolon.
- **#include** weist den Präprozessor an, eine weitere Quelldatei einzulesen.
- stdio.h ist eine Bibliothek, die Funktionen für die Ein- und Ausgabe enthält. Die Datei selber befindet sich im Verzeichnis *Laufwerk:\Programme\Ordner_des_Compilers\Include.*
- **#define** definiert Konstanten.

Das erste Programm

```
/************************
   Filename: hello.c
   Beschreibung: Ausgab
   History :    Date
                04-06
 ********************

#include <stdio.h>

/* Beginn des Pro   ams
int main(void)
 {
    printf("Hello, worl
    return 0;
 }
```

- **main** ist ein Funktionsname. Dies ist die einzige Funktion, die in einem C-Programm vorhanden sein muss.
- **main** ist die erste Funktion, die aufgerufen wird, wenn ein Programm startet.
- Die Funktion selber gibt eine ganze Zahl (**int**) an den Aufrufer zurück. Mit Hilfe dieser Zahl kann eine Fehlerbehandlung durchgeführt werden.
- Der Funktion werden keine Argumente übergeben (**void**). Die Argumente werden immer in runde Klammer gefasst. Sie stehen immer direkt hinter dem Funktionsnamen. Die Klammern können auch leer sein, wenn keine Argumente übergeben werden.
- Die Funktion beginnt mit der geöffneten geschweiften Klammer und endet mit der geschlossenen geschweiften Klammer.

Das erste Programm

R|R|Z|N

```
/*************************************************************
   Filename: hello.c
   Beschreibung: Ausgabe von Hello World
   History :    Date                Reason
                04-06-05            Created
 *************************************************************/

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

- Ein Block beginnt immer mit den geöffneten geschweiften Klammern und endet mit den geschlossenen geschweiften Klammern.
- Innerhalb der Klammern stehen eingerückt Anweisungen.
- Die Einrückungen erhöhen die Lesbarkeit des Programms.
- Jede Anweisung innerhalb eines Blockes endet mit einem Semikolon.

Das erste Programm

```
/*****************************************
   Filename: hello.c
   Beschreibung: Ausgabe von Hello World
   History :    Date              Reason
                04-06-05          Created
 *****************************************/

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

- **printf()** ist eine Funktion, die in der Bibliothek *stdio.h* vordefiniert ist.
- Die Funktion gibt in der MS-Eingabeaufforderungen einen Text aus.
- Der Text wird innerhalb der runden Klammern der Funktion übergeben.
- Hier wird der Text "Hello World!" ausgegeben. Ein Text beginnt und endet immer mit Anführungszeichen.
- \n steht für New Line. Es wird eine Leerzeile ausgegeben.

Das erste Programm

```
/*************************************************************
   Filename: hello.c
   Beschreibung: Ausgabe von Hello World
   History :    Date                Reason
                04-06-05            Created
 *************************************************************/

#include <stdio.h>

/* Beginn des Programms */
int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
```

- **return** gibt einen Wert an die aufrufende Funktion zurück.
- Hier wird Null an die Funktion zurückgegeben.
- In der Funktion ist kein Fehler aufgetreten.

Aufbau eines C-Programms

```
/* Informationen zu dem Programm */

/* Präprozessoranweisung */
#include <datei.h>
#define SYMBOL Text

/* Hauptprogramm */
Rückgabewert main (Argumente)
      {
      Vereinbarung von Variablen
      Anweisungen
      return 0;
      }
```

- Der Programmkopf des C-Programms besteht aus…
  - ... einem Kommentar, der Informationen zu dem Programm enthält und die Aufgabe des Programms beschreibt.
  - ... Präprozessor-Anweisungen, die weitere Dateien einbinden und Platzhalter für Text beinhalten.

Aufbau eines C-Programms

```
/* Informationen zu dem Programm */

/* Präprozessoranweisung */
#include <datei.h>
#define SYMBOL Text

/* Hauptprogramm */
Rückgabewert main (Argumente)
      {
      Vereinbarung von Variablen
      Anweisungen
      return 0;
      }
```

- Jedes C-Programm hat ein Hauptprogramm, welches durch den Bezeichner **main** kennzeichnet wird. Dieses Hauptprogramm kann Werte zurückgeben oder übergeben bekommen.
- Innerhalb des Hauptprogramms können am Anfang Variablen vereinbart werden. Variablen sind Platzhalter für Werte im Speicher eines Rechners. Anschließend folgen Anweisungen, die eine bestimmte Aufgabe lösen.

Programmbeispiel “ggT“ (1)

R|R|Z|N

```
/*****************************************************************************
  Filename: ggt.c
  Beschreibung: Berechnung des größten gemeinsamen Teilers
                zweier ganzer Zahlen
  History : Date          Reason
            04-07-05      Created

*****************************************************************************/
/* Die Header-Datei enthält Funktionen für die Ein- und Ausgabe am
Bildschirm */
#include <stdio.h>

int main ( void)
{
/* Variablendeklaration */
int      zahl1;
int      zahl2;
int      hilf;
int      rest;
int      x;
int      y;

/* Zwei Zahlen werden vom Bildschirm eingelesen */
printf("Eingabe der ersten, ganzen Zahl: ");
scanf("%d", &zahl1);
printf("Eingabe der zweiten, ganzen Zahl: ");
scanf("%d", &zahl2);
```

Programmbeispiel “ggT“ (2)

R|R|Z|N

```
/* Zahlen tauschen, falls erforderlich */
if (zahl1 < zahl2)
{
hilf = zahl1;
zahl1 = zahl2;
zahl2 = hilf;
}

/* wenn zahl2 größer als 0 ist */
if (zahl2 > 0)
{
    /* Siehe nächste Seite */

}
else
{
printf("Ungueltige Eingabe!\n");
}

return 0;
}
```

Sind die Einrückungen sinnvoll gewählt?

Programmbeispiel “ggT“ (3)

R|R|Z|N

```
x = zahl1;
y = zahl2;
rest =  x % y;

/* Berechnung des ggTs */
while (rest != 0 )
      {
       rest = x % y;
       x = y;
       y = rest;
       printf("Ausgabe der Einzelschritte: ggt(%d, %d) = %d\n", x, y, rest);
                }

       /* Ausgabe ggT */
       printf("ggt(%d, %d) = %d\n", zahl1, zahl2, x);
      }
```

Sind die Einrückungen sinnvoll gewählt?

Programmbeispiel “ggT“ (4)

- Dieses Programm spiegelt den Ablaufplan für das Problem "Finde den größten gemeinsamen Teiler zweier ganzer Zahlen" wieder.
- Die Funktion **scanf()** ist in der Bibliothekt *stdio.h* vordefiniert. Die Funktion liest einen Wert aus der Eingabeaufforderung in eine Variable ein.
- **int zahl1, zahl2, hilf, rest, x, y;** definiert Variablen für die Berechnung. Diese Variablen können durch das Schlüsselwort **int** nur mit ganzen Zahlen rechnen. Was passiert, wenn Sie Zahlen mit Nachkommastellen am Bildschirm eingeben?
- **if (Bedingung)** führt Anweisungen innerhalb eines Blockes nur aus, wenn die Bedingung in den runden Klammern erfüllt ist. **else { }** wird ausgeführt, wenn die Bedingung nicht erfüllt ist.
- **while (Bedingung)** ist eine Schleife, die die Anweisungen innerhalb des Blocks solange ausführt wie die Bedingung innerhalb der runden Klammern erfüllt ist.
- **x = zahl1** weist der Variablen x den Wert aus der Variablen zahl1 zu. **rest = x % y** berechnet den Rest von der Division x durch y.
- **printf("Test: ggt(%d, %d) = %d\n", x, y, rest);. %d** sind Platzhalter für ganzzahlige Variablen, die am Bildschirm ausgegeben werden.

Programmbeispiel “ggT“ (5)

R|R|Z|N

- Geben Sie das Programmbeispiel „ggT“ in Ihr Entwicklungssystem ein
- Fügen Sie Kommentare hinzu.
- Analysieren Sie das Programm und finden Sie mögliche Fehler heraus.
- Führen Sie einen systematischen Test durch, ob das Programm alle Fälle beherrscht.

Layout von Programmen (1)

```
int main(void){int zahl1,zahl2,hilf,rest,x, y;printf("Eingabe der
ersten, ganzen Zahl: ");
scanf("%d",&zahl1);printf("Eingabe der zweiten,", "ganzen Zahl:
");scanf("%d", &zahl2);
if (zahl2 > zahl1){hilf=zahl1;zahl1=zahl2; zahl2=hilf;}if(zahl2>0)
{x=zahl1;y=zahl2;
rest=1;while(rest>0){rest=x%y;x=y;y=rest;
}printf("ggt(%d, %d) = %d\n",zahl1,zahl2,
x);}else{printf("Ungueltige Eingabe!\n"); }system("PAUSE");return
0;}
```

- So versteht es zwar der Compiler, aber sonst wohl kaum jemand.
- Die Fehlermeldungen des Compilers beziehen sich auf eine Zeile des Quellcodes.
- Wenn mehr als ein Befehl in der Zeile steht, kann man den Fehler schlecht finden.

Layout von Programmen (2)

R|R|Z|N

```
int main ( void)
{
int zahl1,zahl2,hilf,rest,x,y;
printf("Eingabe der ersten, ganzen Zahl: ");
scanf("%d", &zahl1);
printf("Eingabe der zweiten, ganzen Zahl: ");
scanf("%d",&zahl2);
if(zahl2>zahl1)
{
hilf=zahl1;
zahl1=zahl2;
zahl2=hilf;
}
```

- Mit einem Befehl pro Zeile sieht es schon besser aus
- Die Struktur des Programms ist aber leider noch immer nicht gut zu erkennen.
- Der Code ist ohne Leerzeichen oder Tabs geschrieben, was die Lesbarkeit erschwert.

Layout von Programmen (3)

R|R|Z|N

```
if (zahl2 > zahl1)
   {
   hilf  = zahl1;
   zahl1 = zahl2;
   zahl2 = hilf;
   }
i = i + 1;
```

```
if (zahl2 > zahl1){
  hilf  = zahl1;
  zahl1 = zahl2;
  zahl2 = hilf;
     }
i = i + 1;
```

- Durch Einrückungen wird die Struktur des Programms deutlich.
- Ideale Einrückung 2-4 Zeichen.
- Es haben sich die beiden oben genannten Stile für die Formatierung von Einrückungen etabliert.
- Zusätzliche Leerzeichen verbessern die Lesbarkeit.

Bezeichner

R|R|Z|N

- ... sind eine Folge von Buchstaben und Ziffern, die mit einem Buchstaben beginnen und als Sonderzeichen einen Unterstrich enthalten können.
- ... sind Namen von Funktionen (Modulen), Variablen und Konstanten in C.
- Ein Schlüsselwort ist ein von C reservierter Bezeichner, der nicht anders verwendet werden darf.

Der ANSI-Standard kennt folgende Schlüsselwörter:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while