



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

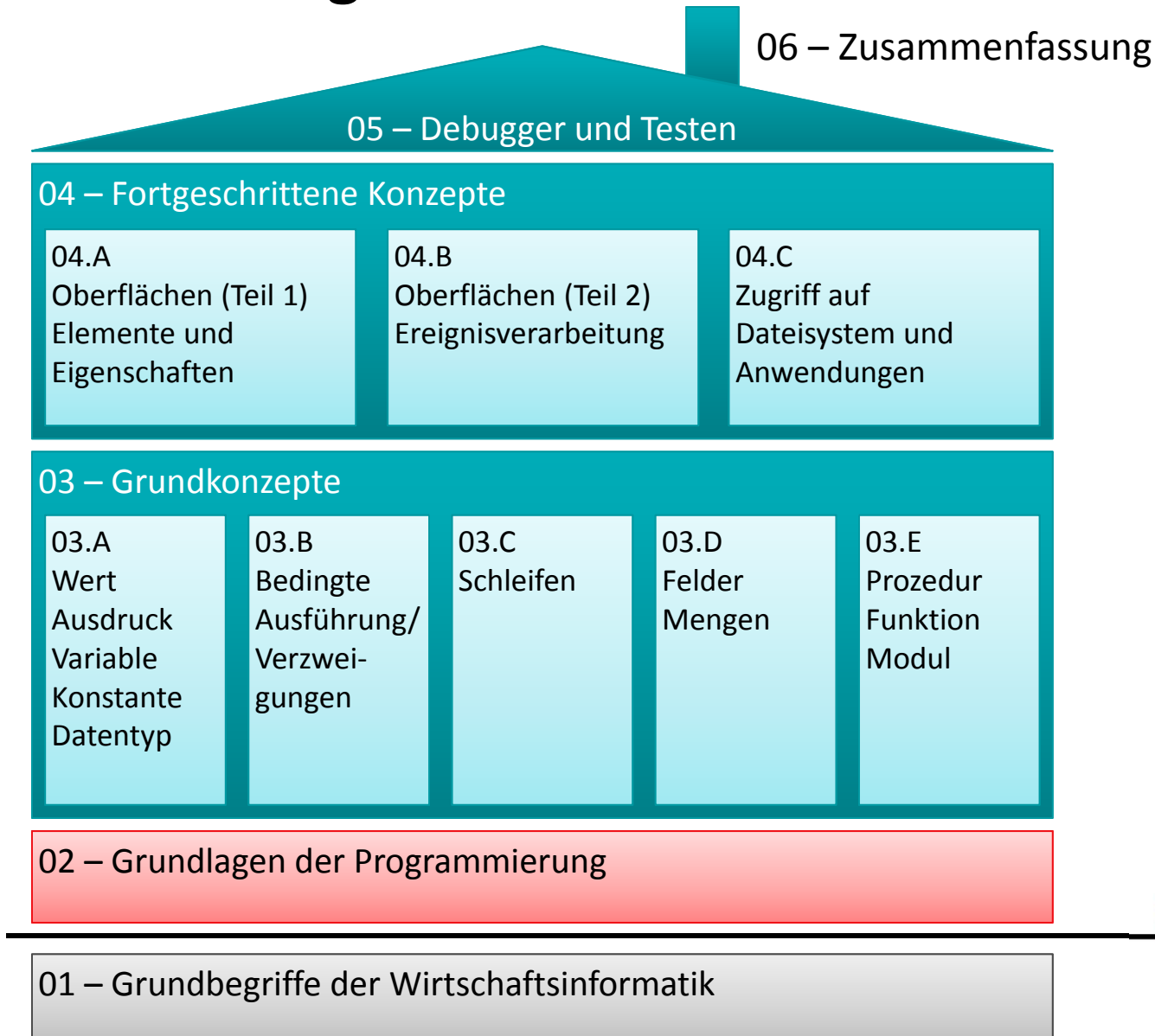
Wirtschaftsinformatik 1

LE 02 – Grundlagen der Programmierung

Prof. Dr. Thomas Off

<http://www.ThomasOff.de/lehre/beuth/wi1>

Einordnung





Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick

Rückblick



Wirtschaftsinformatik

- als interdisziplinäre, anwendungsorientierte und gestaltungsorientierte Wissenschaft, deren Erkenntnisgegenstand soziotechnische Systeme sind
- umfasst auch Konzeption, Entwicklung, Einführung, Wartung und Nutzung der computergestützten Verarbeitung von Informationen für betriebswirtschaftliche Aufgaben in Wirtschaftsunternehmen und unternehmensübergreifenden Netzen
- sowie zunehmend deren Management und Innovation

Rückblick

Grundbegriffe

- Information, Daten, Wissen
- Computer, EVA-Prinzip und Hardware
- Software
- System und Modell



Rückblick

Grundbegriffe

- Information, Daten, Wissen

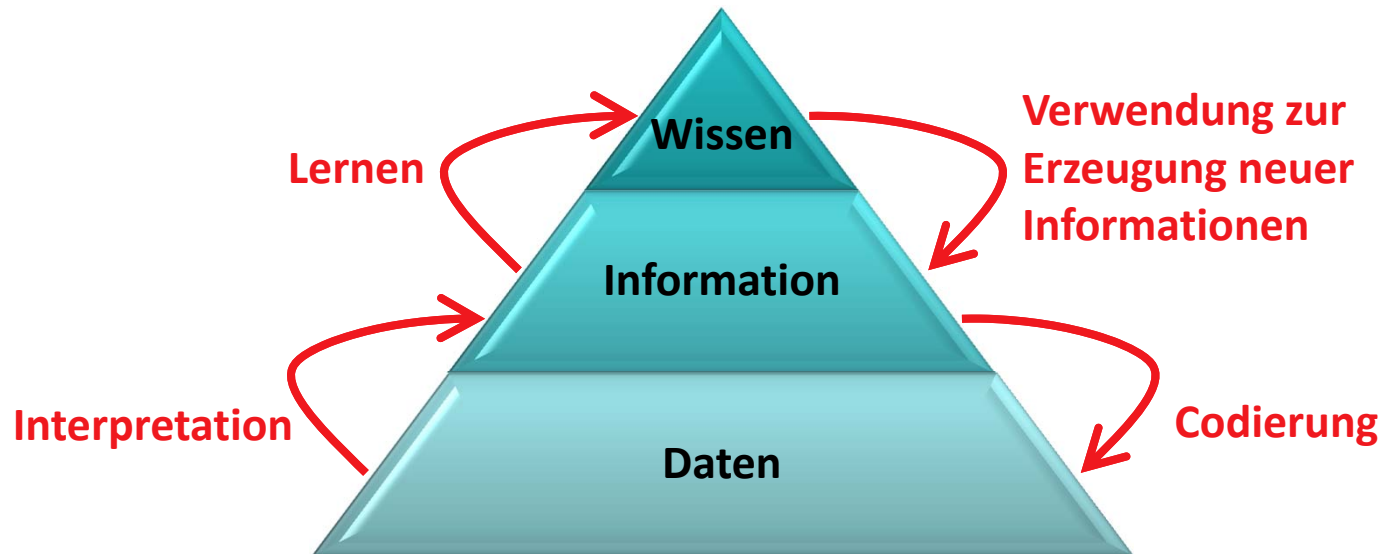


Abb. nach [Fink&Schneiderei, 2001], S. 69

- Computer, EVA-Prinzip und Hardware
- Software
- System und Modell

Rückblick

Grundbegriffe

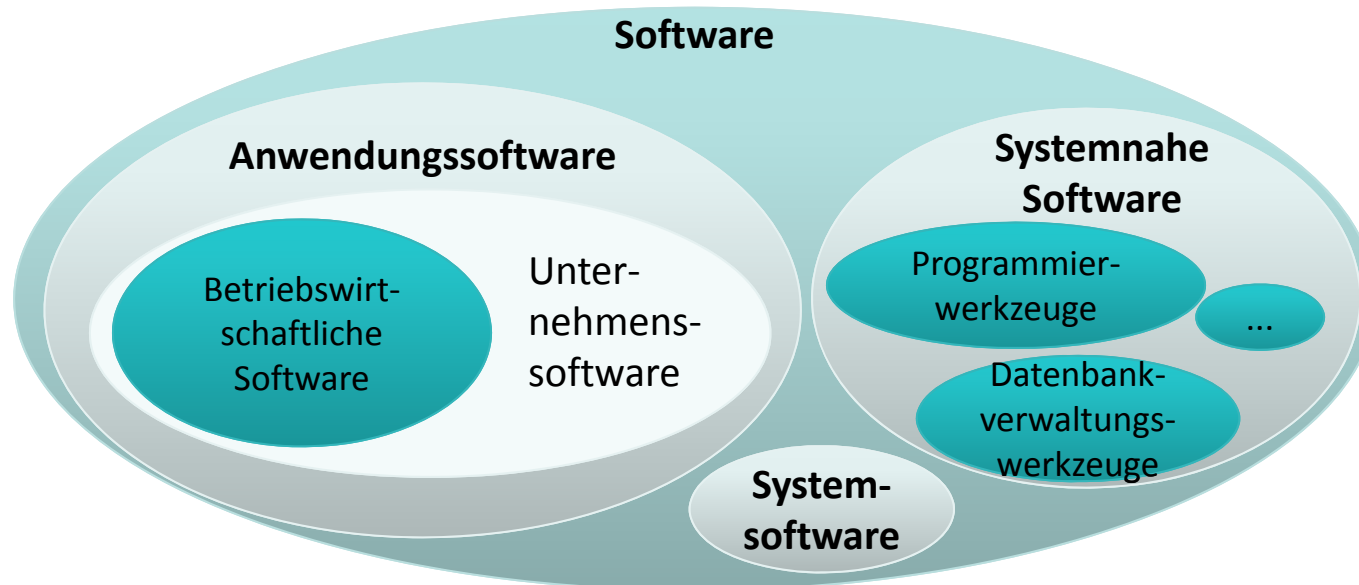


- Information, Daten, Wissen
- Computer, EVA-Prinzip und Hardware
 - Universell einsetzbare, programmgesteuerte Maschine zur Speicherung und Verarbeitung von Daten
 - Erklärung anhand von Eingabe (E), Verarbeitung (V) und Ausgabe(A) mit entsprechenden Hardwarekomponenten (physische Teile)
 - Wichtige Komponenten
 - CPU zur Verarbeitung, d.h. zur Steuerung der abzuarbeitenden Verarbeitungsvorschriften und zur Ausführung von Rechenoperationen
 - Arbeitsspeicher für Befehle und Daten
 - Von-Neumann-Rechner als Grundlegendes Architekturprinzip
- Software
- System und Modell

Rückblick

Grundbegriffe

- Information, Daten, Wissen
- Computer, EVA-Prinzip und Hardware
- Software



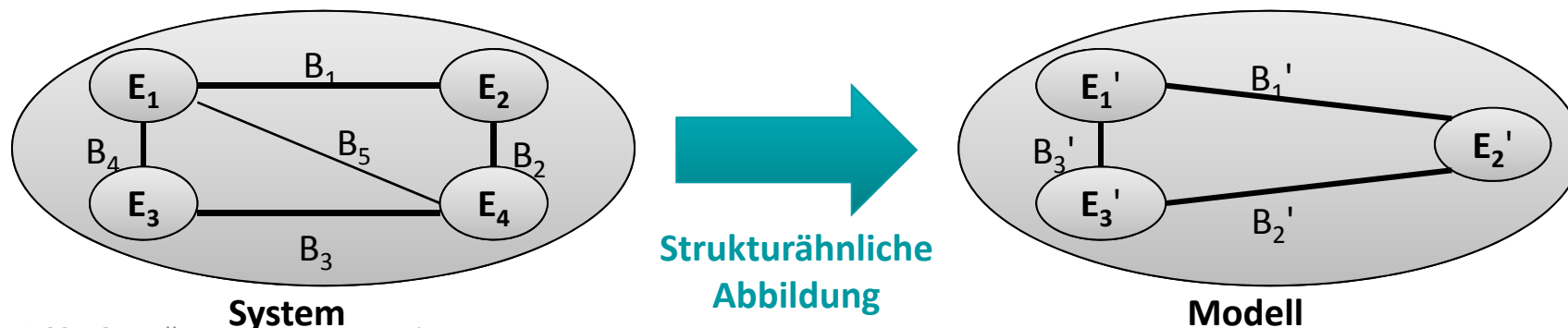
- System und Modell

Rückblick

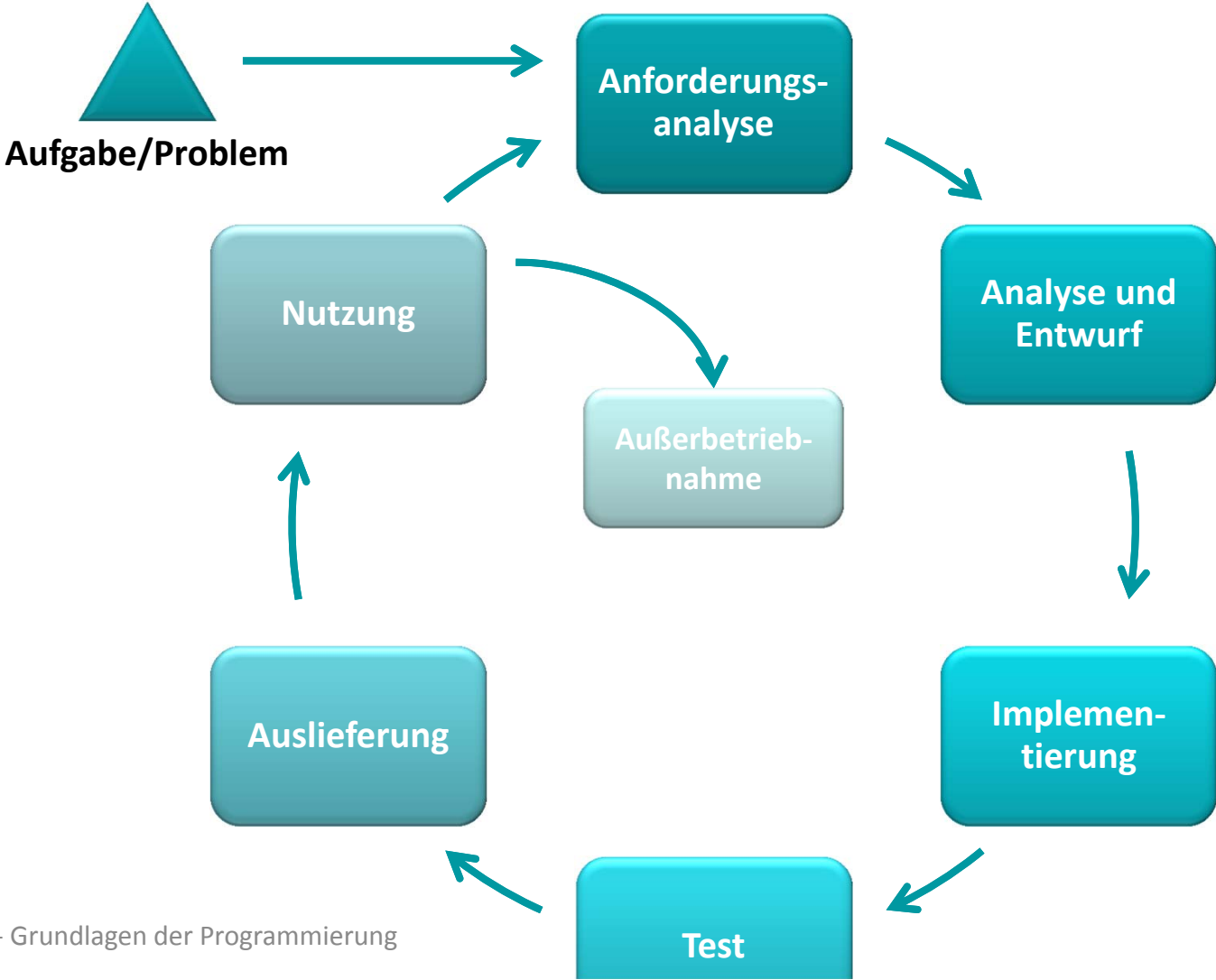
Grundbegriffe



- Information, Daten, Wissen
- Computer, EVA-Prinzip und Hardware
- Software
- System und Modell
 - System als Menge von Elementen, die durch eine Menge von Beziehungen (Relationen) miteinander verbunden sind.
 - "Ein Modell ist ein abstraktes System, das ein anderes (meist reales) System [für einen bestimmten Zweck] in vereinfachter Weise abbildet." [Krallmann, 1996], S. 12 f.



Rückblick Softwarelebenszyklus





Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

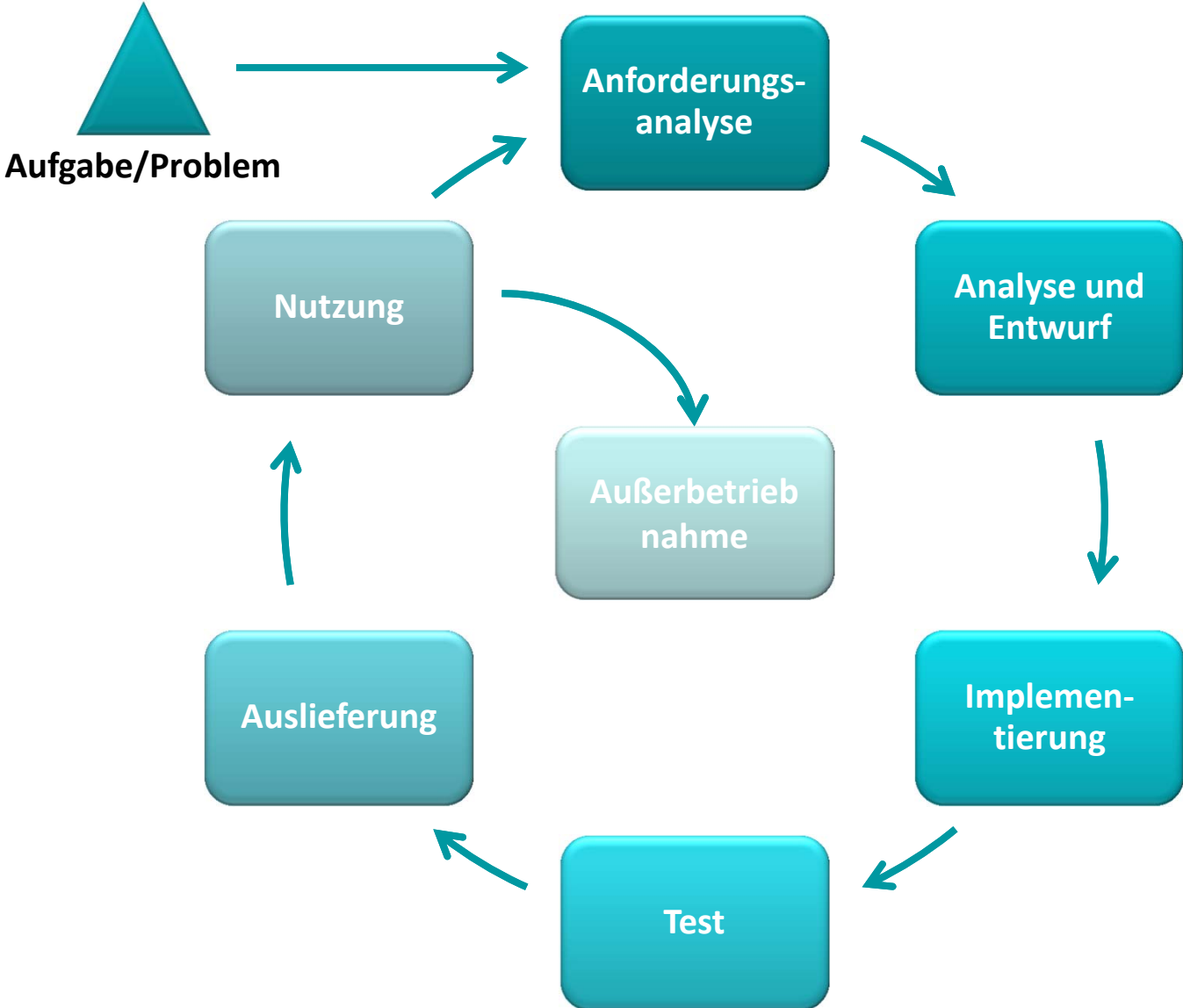
MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

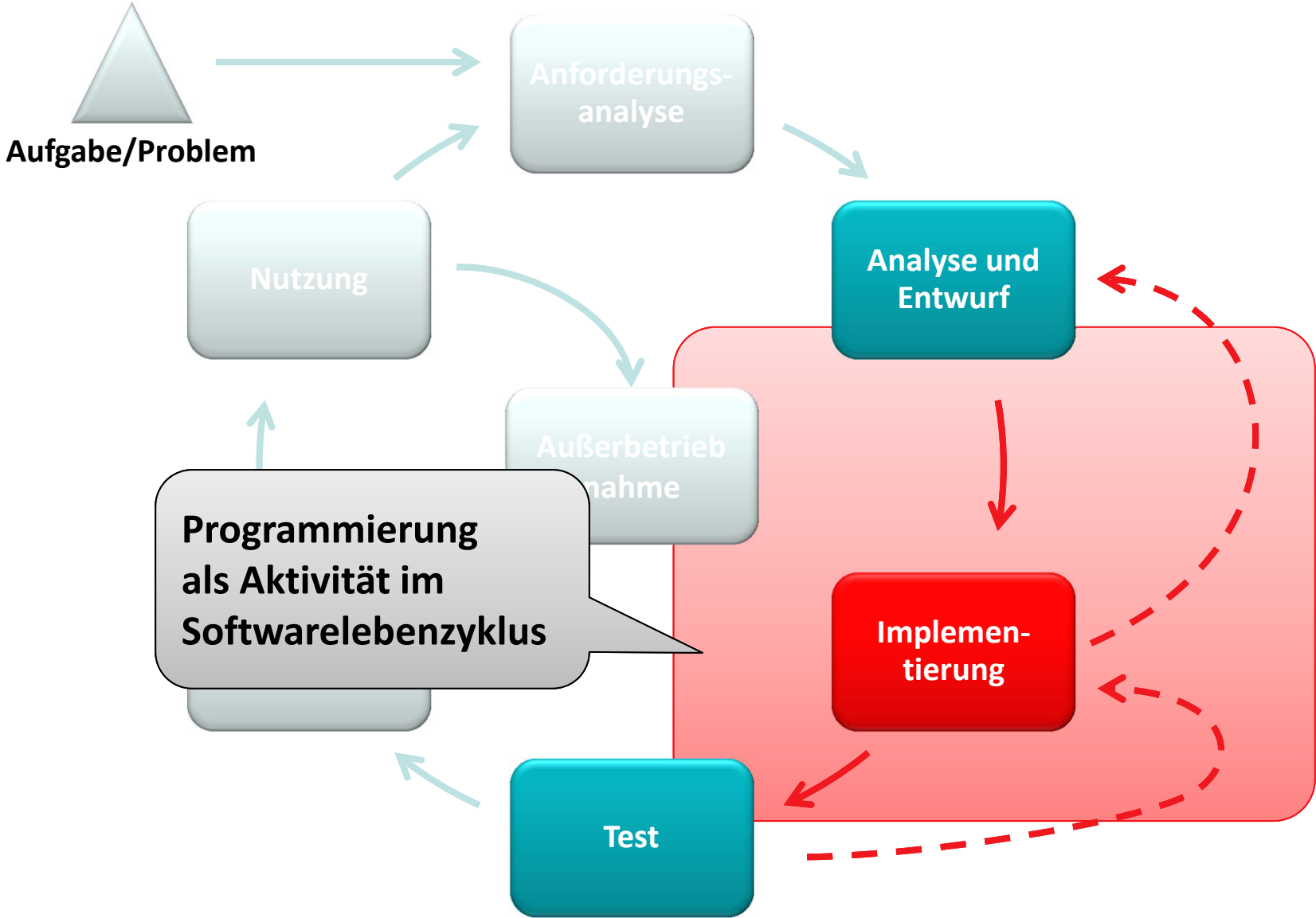
Zusammenfassung

Ausblick

Einordnung



Softwarelebenszyklus (Überblick)



Softwarelebenszyklus (Überblick)



**Programmierung
als Aktivität im
Softwarelebenszyklus**



Computergestützte Verarbeitung von Informationen



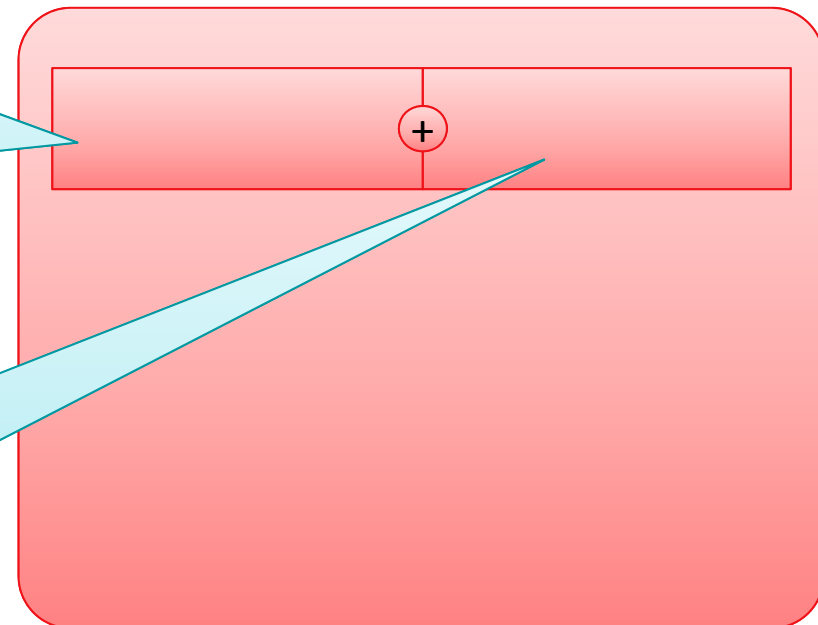
Wenn Gegenstand der Informatik die

Computergestützte Verarbeitung von Informationen

ist, dann geht es um¹

Verarbeitung in Form von
Verarbeitungsvorschriften,
ihrer Bestandteile und
Darstellungsform

Repräsentation von
Informationen als Daten,
Datenstrukturen und
ihre Beziehungen
untereinander



1) vgl. [Broy, 1992], S. 3 und übereinstimmend [Lehner et al., 2008], S. 139 ff.

Algorithmus, Datenstruktur und Programm

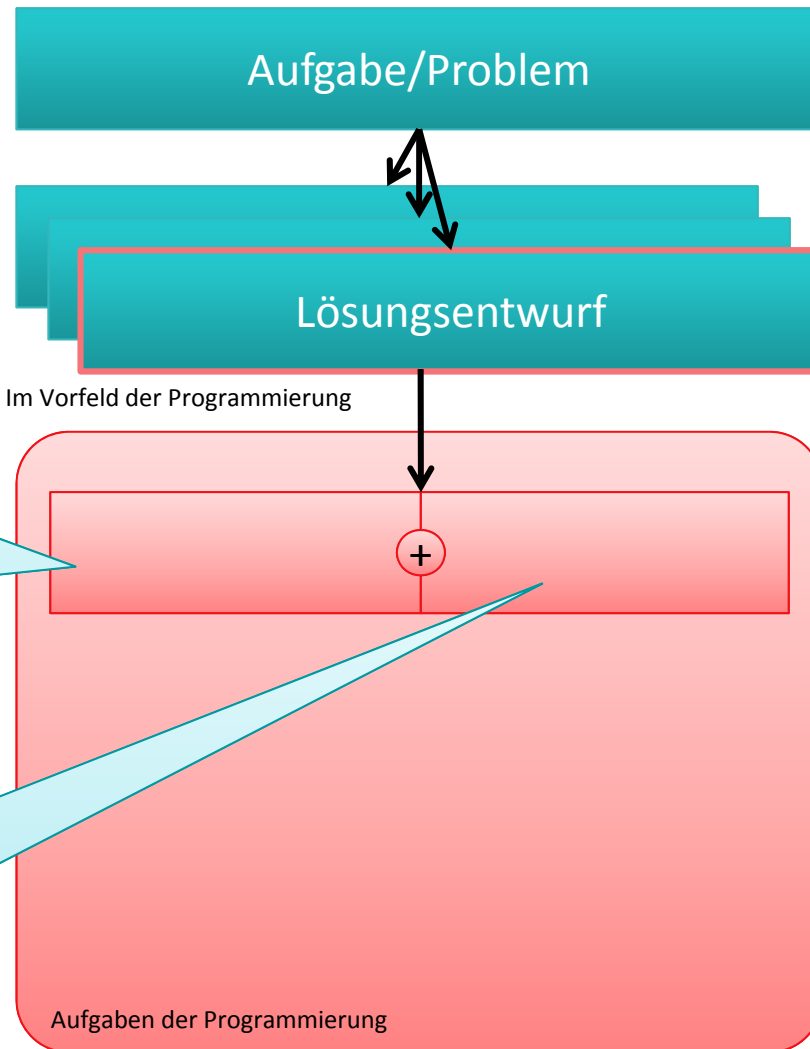


Im Vorfeld

- Aufgabe/Problem analysiert
- mindestens eine Lösung für Problem entworfen

Verarbeitung in Form von Verarbeitungsvorschriften, ihrer Bestandteile und Darstellungsform

Repräsentation von Informationen als Daten, Datenstrukturen und ihre Beziehungen untereinander

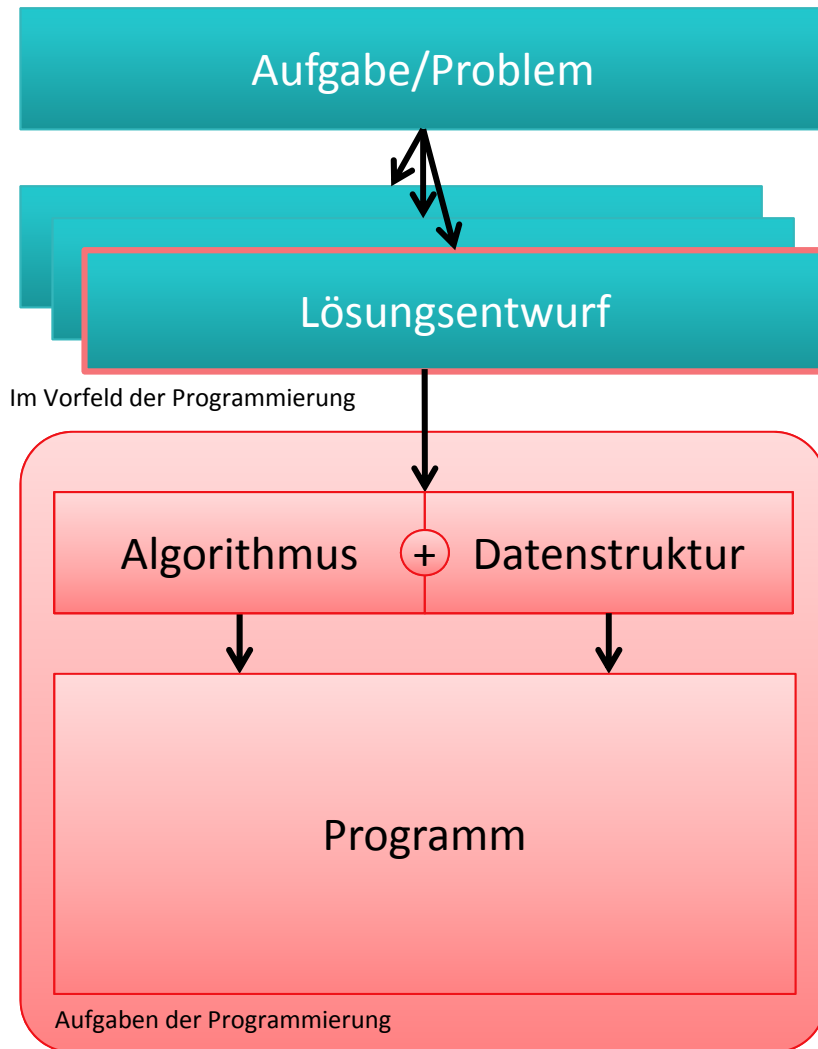


Algorithmus, Datenstruktur und Programm



Stufenweise Umsetzung der Lösung

- Algorithmus: spezielle Form der Verarbeitungsvorschrift
- Datenstruktur: Aufbau und Organisation der zu verarbeitenden Daten
- Programm: Umsetzung eines Algorithmus mit Mitteln einer Programmiersprache und Ausführung durch Befehle aus dem Befehlsvorrat des Computers (Maschinencode)

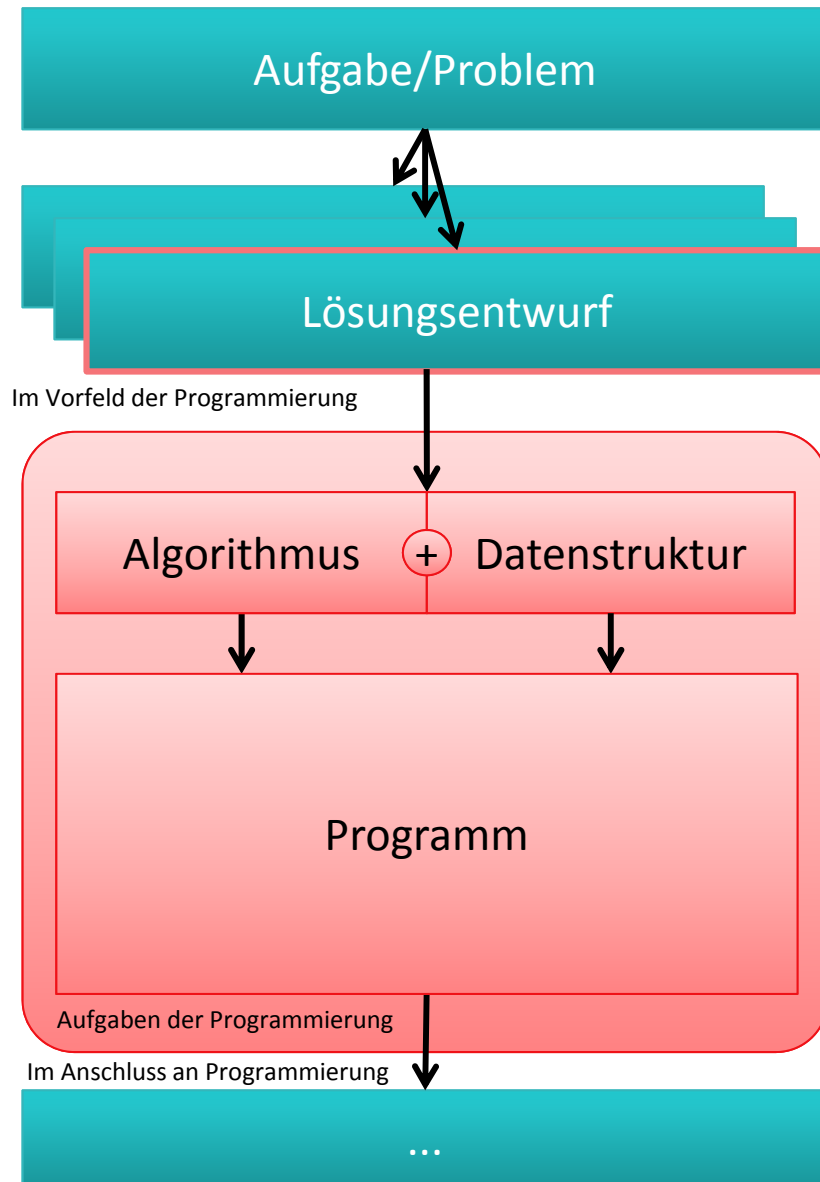


Algorithmus, Datenstruktur und Programm

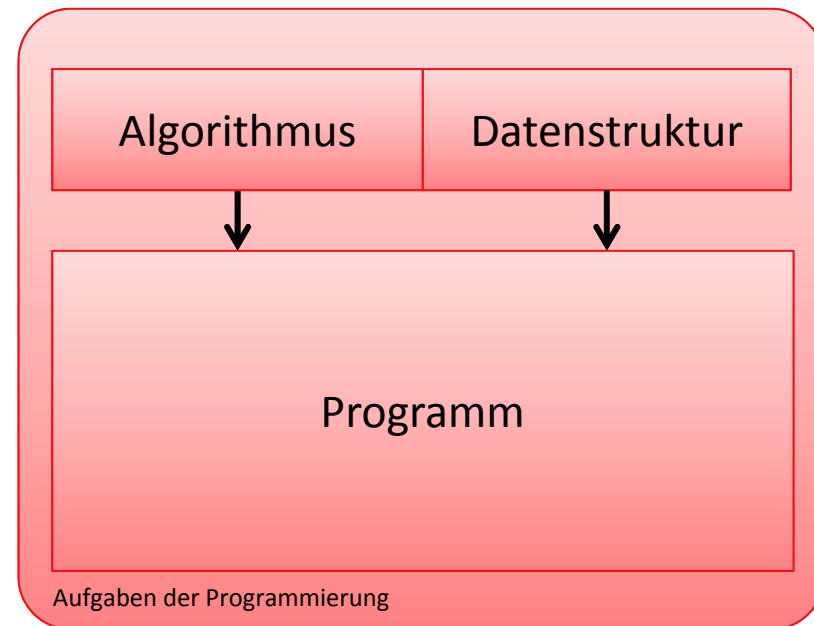


Im Anschluss

- ausführbares Programm
- bereit für folgende Aktivitäten
 - z.B. zum Test



Algorithmus, Datenstruktur und Programm

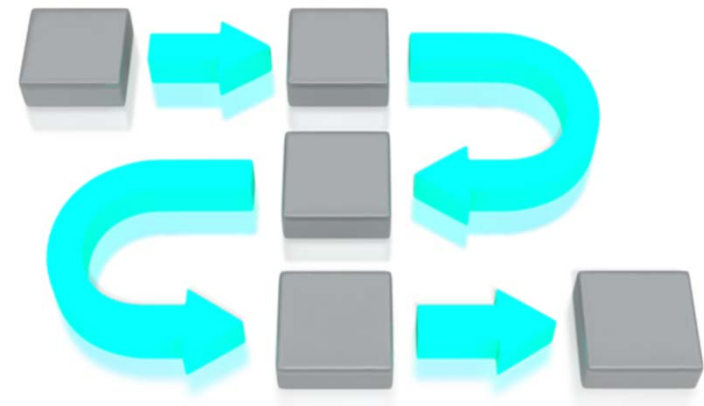


Algorithmus



Definition Algorithmus:¹

- präzise und vollständig in einer eindeutigen Sprache formulierte Verarbeitungsvorschrift,
- die eine endliche Abfolge einzeln ausführbarer Verarbeitungsschritte vorgibt,
- die eine Ausgangssituation in ein Ergebnis überführen,
- das zur Lösung einer Aufgabenstellung/eines Problems dienen soll.



vgl. hier und im Folgenden [Duden, 2001], S. 43 ff.

Beispiel "Kaffeekochen"



1. Gegeben sind eine einsatzbereite Kaffeemaschine, Kaffeepulver, Wasser und Filtertüten
2. Kanne mit 300 ml Wasser füllen
3. Wasser in den Wasserbehälter füllen
4. Kanne unter den Filter stellen
5. Filtertüte in den Filter einsetzen
6. 3 Löffel Kaffeepulver in den Filter geben
7. Kaffeemaschine einschalten
8. Warte etwas
9. Prüfe ob Wasser durch den Filter gelaufen und die Kaffeekanne gefüllt ist. Wenn nicht wiederhole 8 bis 9.
10. Maschine ausschalten
11. Maschine in einsatzbereiten Zustand bringen (z.B. Filter entleeren, ggf. reinigen)



Quelle: [1]

Beispiel "Zubereitung einer Tiefkühlpizza"



1. Gegeben sind Tiefkühlpizzen im Tiefkühlfach, Backofen/-blech
2. Schalte den Backofen ein und stelle ihn auf 230 Grad
3. Nimm die Pizzapackung aus dem Tiefkühlfach
4. Öffne die Packung
5. Entferne die Folie von der Pizza
6. Lege die Pizza auf ein Backblech
7. Schiebe das Backblech in den Backofen
8. Prüfe, ob Du größeren Hunger hast und ob noch Pizza im Tiefkühlfach ist und ob noch Platz im Backofen ist. Wenn ja, dann wiederhole 3 bis 8. Andernfalls warte etwas.
9. Prüfe, ob die Pizza braun geworden und/oder der Käse verlaufen ist. Wenn nicht, dann warte etwas und anschließend weiter bei 9. Andernfalls nimm alle Backbleche heraus und stelle sie ab.
10. Schließe den Backofen
11. Schalte den Backofen aus



Beispiel "Zubereitung von Spiegeleiern"



1. Gegeben sind Eier in einer Packung, Pfanne mit Deckel, Teller, Herd, Fett, Salz, Pfeffer, Pfannenwender, Mülleimer
2. Stell eine Pfanne auf den Herd und einen Teller neben den Herd
3. Gib etwas Fett in die Pfanne
4. Schalte den Herd auf mittlerer Stufe ein
5. Nimm ein Ei aus der Packung und schlage es auf
6. Gib das Eiweiß und Eigelb in die Pfanne
7. Wirf die Eierschale in den Müll
8. Prüfe, ob Du größeren Hunger hast und noch Eier in der Packung sind. Dann wiederhole vorherige drei Schritte (5 bis 8).
9. Streue etwas Salz und Pfeffer auf die Eier
10. Setze den Deckel auf die Pfanne.
11. Prüfe, ob das Eigelb fest geworden ist. Wenn nicht, warte etwas und wiederhole 11.
12. Andernfalls nimm einen Pfannenwender.
13. Hebe die Eier mit dem Pfannenwender aus der Pfanne auf den Teller.
14. Schalte den Herd ab



Quelle: [2]

Beispiel "Währungsumrechnung EUR in USD"



1. Gegeben ist
 - a) der Umrechnungsfaktor U von EUR in USD und
 - b) ein Währungsbetrag E in EUR
2. Multipliziere E mit U
3. und merke das Ergebnis im Währungsbetrag D
4. Gibt den Währungsbetrag D als Ergebnis in USD



Beispiel "Fakultät n!"



Hinweise

- Formel: $f(n) = n! = n * n-1 * \dots * 2 * 1$; per Definition $f(0) = 1$
- Beispiele: $f(3) = 3! = 3*2*1 = 6$, $f(4) = 4! = 4*3*2*1 = 24$

Algorithmus

1. Gegeben ist eine positive Zahl n
2. Speichere 1 im Ergebnis e
3. Wenn n größer 0 ist
 - a) Multipliziere e mit n und speichere das Ergebnis in e
 - b) Reduziere n um eins
 - c) Mache weiter bei 3.
4. Gib e als Ergebnis

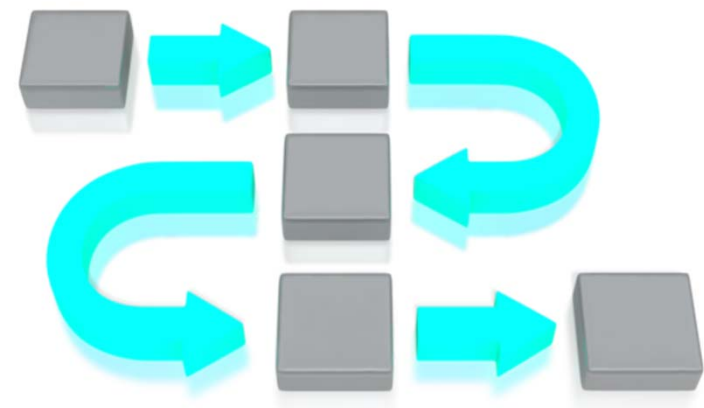


Algorithmus



Definition Algorithmus:¹

- präzise und vollständig in einer eindeutigen Sprache formulierte Verarbeitungsvorschrift,
- die eine endliche Abfolge einzeln ausführbarer Verarbeitungsschritte vorgibt,
- die eine Ausgangssituation in ein Ergebnis überführen,
- das zur Lösung einer Aufgabenstellung/eines Problems dienen soll.



vgl. hier und im Folgenden [Duden, 2001], S. 43 ff.

Informelle Algorithmenbeschreibung



Vorangegangenen Beispiele scheinen für Menschen Algorithmen zu sein,

- weil sie möglichst präzise und vollständig in natürlicher Sprache formuliert sind
- setzen aber gewisse Grundkenntnisse voraus (Temperatur des Backofens einstellen, Ei aufschlagen, Multiplikation, ...)
- und lassen Spielraum für Interpretation (Pizza braun geworden und/oder der Käse verlaufen , Streue etwas Salz und Pfeffer, Gib Ergebnis ...)

Formale Algorithmenbeschreibung



Wenn ein Computer den Algorithmus ausführen soll,

- müssen die zur Formulierung benutzten Sprachelemente auf Elemente einer Programmiersprache bzw. auf den Befehlsumfang des Computers abgebildet werden können
- und der Algorithmus sollte
 - terminieren, d.h. nach endlich vielen ausgeführten Schritten zu einem Ergebnis kommen
 - determiniert sein, d.h. für eine gleiche Ausgangssituationen auch stets gleiche Ergebnisse liefern
 - deterministisch sein, d.h. an jeder Stelle seiner Ausführung höchstens eine Fortsetzungsmöglichkeit haben. Es darf keine Stelle geben, an der nach Belieben ausgewählt werden kann
 - ...

Wichtige Bestandteile von Algorithmen



Elementare Anweisungen

- können bzw. müssen nicht weiter verfeinert werden
 - "Nimm ein Ei aus der Packung"
 - "Speichere 1 in e"

Ausführungsreihenfolge

- sequentiell: Anweisungen nacheinander in der Reihenfolge ausgeführt, in der sie im Algorithmus beschrieben sind
- parallel: Anweisungen werden unabhängig von einander parallel ausgeführt
 - "Während Du die Kanne mit Wasser füllst, setze ich den Kaffeefilter ein."

Wichtige Bestandteile von Algorithmen



Bedingte Ausführung bzw. Verzweigung

- Wenn eine Bedingung zutrifft, führe die Anweisung aus
 - "Wenn n nicht gleich 0 ist ..."
- Wenn eine Bedingung zutrifft, führe die Anweisung aus. Andernfalls führe eine andere Anweisung aus.
 - "Prüfe, ob das Eigelb fest geworden ist. Wenn nicht, warte etwas. Andernfalls nimm einen Pfannenwender."

Schleife

- Wiederholung einer oder mehrerer Anweisungen bis eine bestimmte Anzahl an Wiederholungen erfolgt ist oder eine Bedingung zutrifft
 - "Wenn Du größeren Hunger hast, wiederhole Schritte 5-8."
 - "Solange n noch nicht 0 ist, wiederhole ..."
- Endlosschleifen müssen definitionsgemäß vermieden werden

Wichtige Bestandteile von Algorithmen



Unterprogramm

- Teilvorschrift, die ein sinnvolles Zwischenergebnis produziert und ggf. an mehreren Stellen im Algorithmus verwendet werden kann
- Am Ende der Teilvorschrift wird bei der nächsten Anweisung im Algorithmus fortgefahren, in den die Teilvorschrift eingebunden ist.
- Beispiel "Wirf es weg"
 - "Öffne den Mülleimer, wirf es hinein und schließe den Mülleimer." könnte verwendet werden bei
 - Kaffee kochen: "Wirf den Kaffeefilter weg"
 - Pizza zubereiten: "Wirf die leere Packung weg" oder "Wirf die verbrannte Pizza weg".

Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

Programmablaufplan

UML Aktivitätsdiagramm

...

Darstellungsformen von Algorithmen



Pseudocode

- umfasst Sprachelemente mit einer vorher verabredeten Bedeutung, mit denen die Bestandteile von Algorithmen ausgedrückt werden können
- unabhängig von konkreter Programmiersprache, jedoch häufig Grundelemente von Programmiersprachen angelehnt
- wird eingesetzt, wenn ein "Algorithmus erklärt werden soll und Einzelheiten der Umsetzung in eine Programmiersprache stören würden"¹

Struktogramm (Nassi-Shneiderman-Diagramm)

Programmablaufplan

UML-Aktivitätsdiagramm

1) Quelle: [3]

Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

- Verwendung von rechteckigen Symbolen, um Bestandteile von Algorithmen und Abfolge von Anweisungen auszudrücken
- Entwickelt 1972/73 und normiert in DIN 66261
- Geringe Relevanz in der Praxis, aber ...
- "Erstellung von Struktogrammen ist immer noch Bestandteil vieler [...] Abschlussprüfungen."¹

Programmablaufplan

UML-Aktivitätsdiagramm



Isaac Nassi

Quelle: [6]



Ben Shneidermann

Quelle: [5]

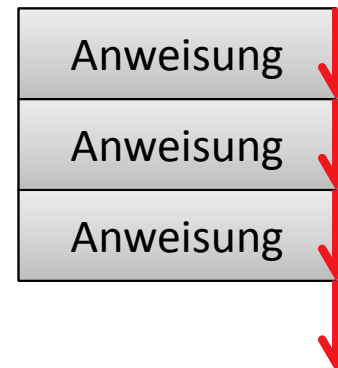
1) Quelle: [4]

Darstellungsformen von Algorithmen



Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge



Darstellungsformen von Algorithmen

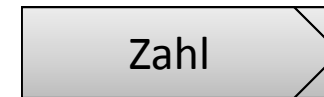


Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge
- Eingabe und Ausgabe



Eingabe einer Zahl



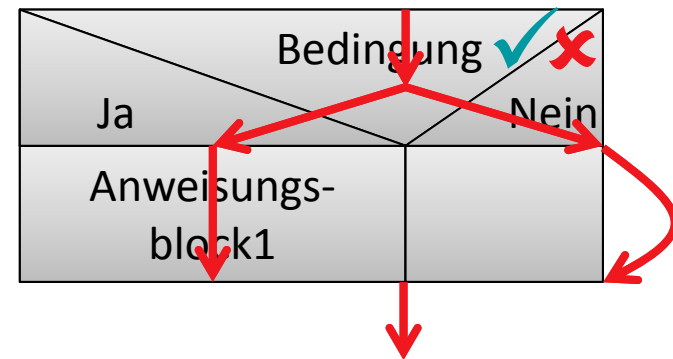
Ausgabe einer Zahl

Darstellungsformen von Algorithmen



Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge
- Eingabe und Ausgabe
- Bedingung/Verzweigung
 - Wenn Bedingung zutrifft, dann führe Anweisungsblock 1 aus.

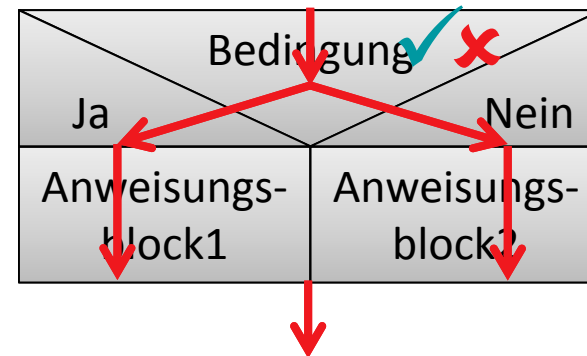


Darstellungsformen von Algorithmen



Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentiell Abfolge
- Eingabe und Ausgabe
- Bedingung/Verzweigung
 - Wenn Bedingung zutrifft, dann führe Anweisungsblock 1 aus, andernfalls Block 2.





Darstellungsformen von Algorithmen

Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge
- Eingabe und Ausgabe
- Bedingung/Verzweigung
 - Abhängig vom Wert der Variablen führe den zugehörigen Anweisungsblock aus.

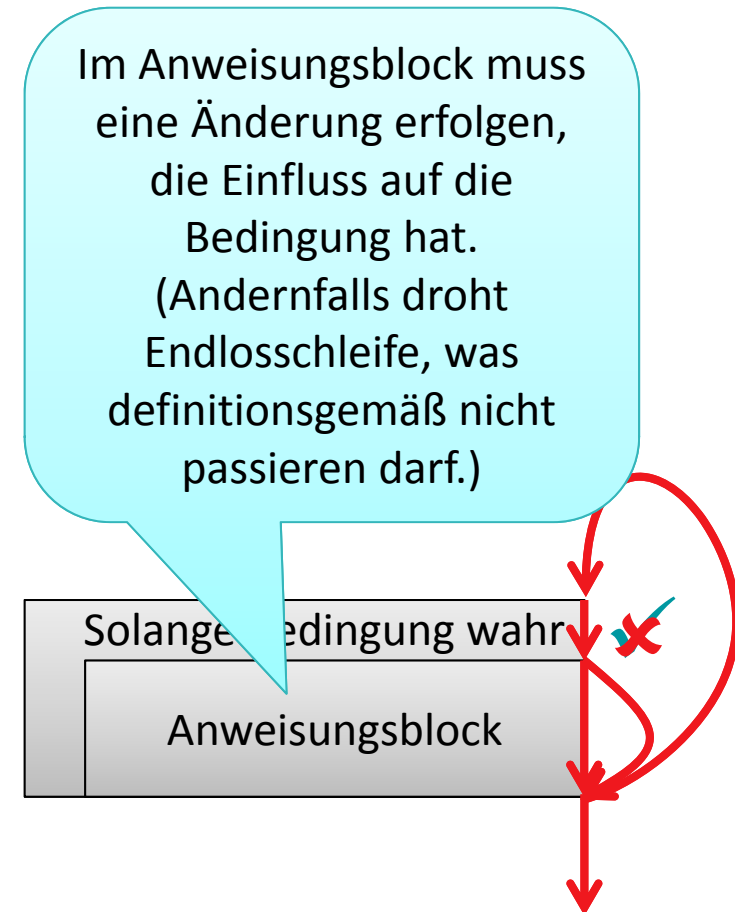
Variable		
Wert 1	Wert 2	Sonst
Anw.-block1	Anweisungs-block2	Anw.-block 3

Darstellungsformen von Algorithmen



Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge
- Eingabe und Ausgabe
- Bedingung/Verzweigung
- Schleife
 - Prüfe die Bedingung und führe den Anweisungsblock nur solange aus, wie die Bedingung zutrifft

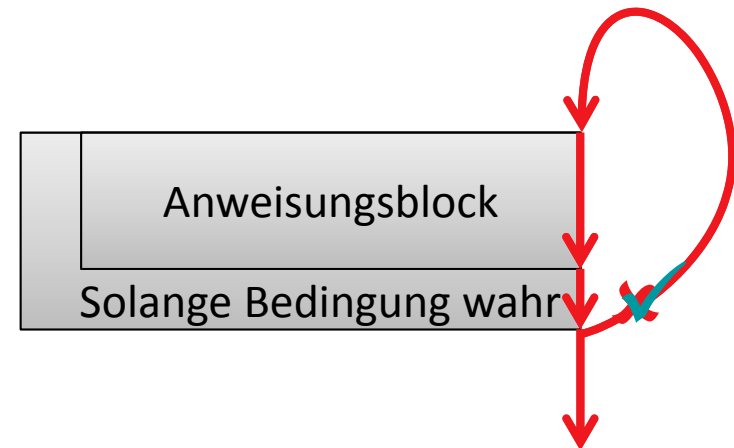


Darstellungsformen von Algorithmen



Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge
- Eingabe und Ausgabe
- Bedingung/Verzweigung
- Schleife
 - Führe den Anweisungsblock aus und wiederhole ihn solange, wie die Bedingung zutrifft

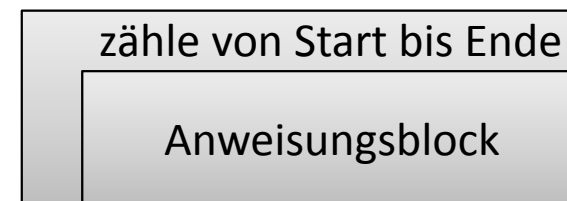


Darstellungsformen von Algorithmen



Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge
- Eingabe und Ausgabe
- Bedingung/Verzweigung
- Schleife
 - Führe eine gegebene Anzahl von Wiederholungen des Anweisungsblocks durch

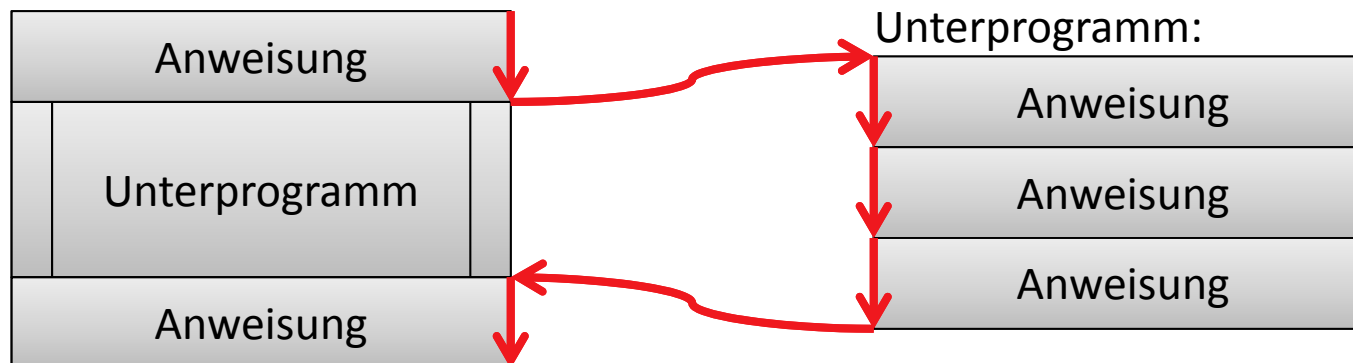




Darstellungsformen von Algorithmen

Struktogramm (Nassi-Shneiderman-Diagramm)

- Anweisung und ihre sequentielle Abfolge
- Eingabe und Ausgabe
- Bedingung/Verzweigung
- Schleife
- Fortsetzung in einem Unterprogramm

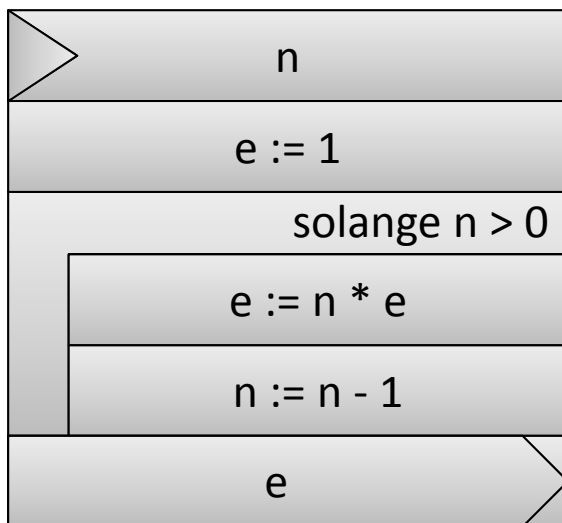


Darstellungsformen von Algorithmen



Beispiel Struktogramm (Nassi-Shneiderman-Diagramm)

– Frage: Welcher Algorithmus ist es?



– Fakultätsberechnung $n!$

1. Gegeben ist eine positive Zahl n
2. Speichere 1 im Ergebnis e
3. Wenn n größer 0 ist
 - a) Multipliziere e mit n und speichere das Ergebnis in e
 - b) Reduziere n um eins
 - c) Mache weiter bei 3
4. Gib e als Ergebnis

Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

- Verwendung von rechteckigen Symbolen, um Bestandteile von Algorithmen und Abfolge von Anweisungen auszudrücken
- Entwickelt 1972/73 und normiert in DIN 66261
- Geringe Relevanz in der Praxis, aber ...
- "Erstellung von Struktogrammen ist immer noch Bestandteil vieler [...] Abschlussprüfungen."¹



Isaac Nassi

Quelle: [6]

Programmablaufplan

UML-Aktivitätsdiagramm



Ben Shneidermann

Quelle: [5]

1) Quelle: [4]

Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

Programmablaufpläne

- durch Pfeile verbundene grafische Symbole stellen den möglichen Ablauf eines Algorithmus dar
- Standardisiert in DIN 66001/ISO 5807
- verlieren in ihrer ursprünglichen Form ihre Praxisrelevanz, da sie zunehmend in Form von UML Aktivitätsdiagrammen Verbreitung finden

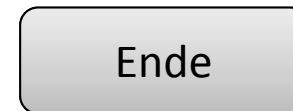
Aktivitätsdiagramm

Darstellungsformen von Algorithmen



Programmablaufpläne

- Häufig verwendete Elemente
 - Start und Ende

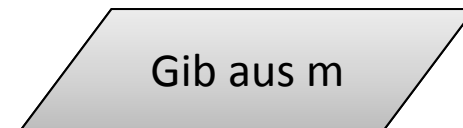


Darstellungsformen von Algorithmen



Programmablaufpläne

- Häufig verwendete Elemente
 - Start und Ende
 - Eingabe und Ausgabe



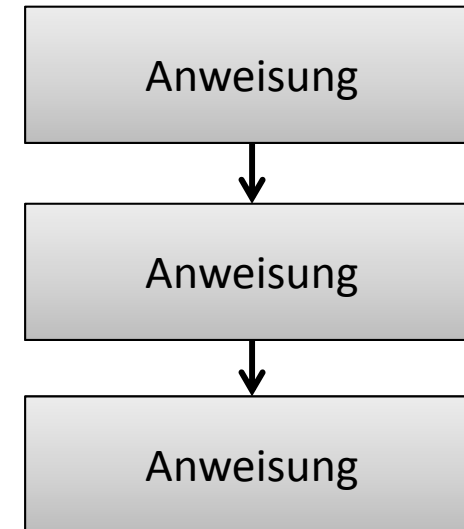
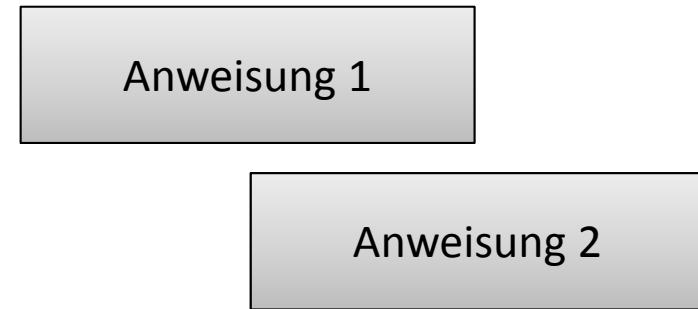
Darstellungsformen von Algorithmen



Programmablaufpläne

– Häufig verwendete Elemente

- Start und Ende
- Eingabe und Ausgabe
- Anweisung und deren sequentielle Abfolge

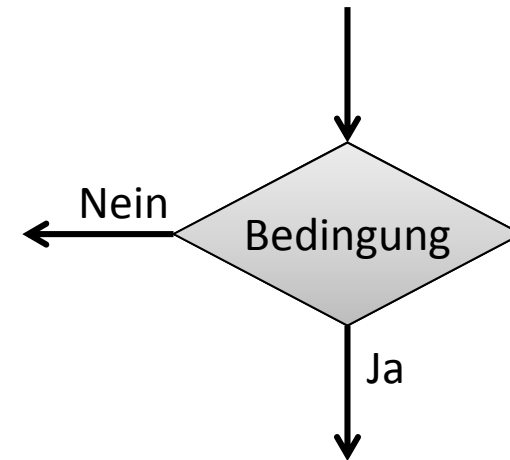


Darstellungsformen von Algorithmen



Programmablaufpläne

- Häufig verwendete Elemente
 - Start und Ende
 - Eingabe und Ausgabe
 - Anweisung und deren sequentielle Abfolge
 - Bedingung/Verzweigung

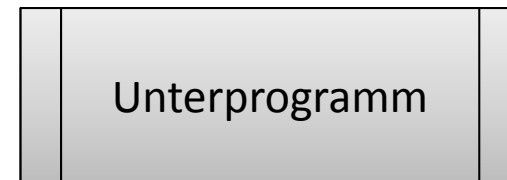


Darstellungsformen von Algorithmen



Programmablaufpläne

- Häufig verwendete Elemente
 - Start und Ende
 - Eingabe und Ausgabe
 - Anweisung und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Unterprogramm



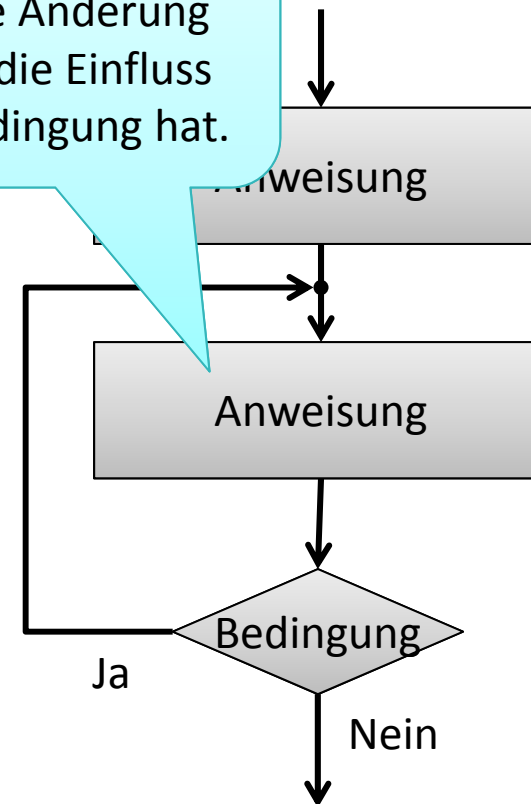
Darstellungsformen von Algorithmen



Programmablaufpläne

- Häufig verwendete Elemente
 - Start und Ende
 - Eingabe und Ausgabe
 - Anweisung und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Unterprogramm
- Kein spezielles Element für Schleifen, stattdessen durch zurücklaufende Pfeile und Bedingungen

Im Anweisungsblock muss eine Änderung erfolgen, die Einfluss auf die Bedingung hat.



Darstellungsformen von Algorithmen



Programmablaufpläne

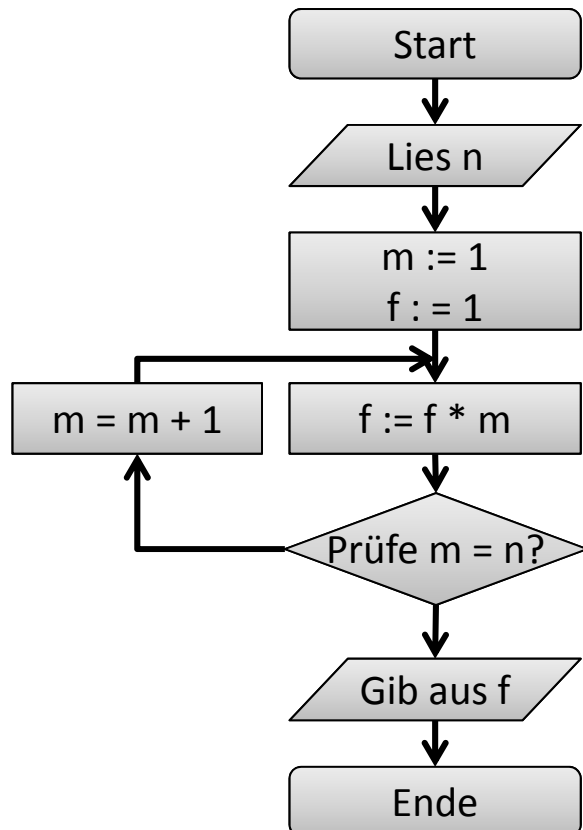
- Häufig verwendete Elemente
 - Start und Ende
 - Eingabe und Ausgabe
 - Anweisung und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Unterprogramm
- Kein spezielles Element für Schleifen, stattdessen durch zurücklaufende Pfeile und Bedingungen

Darstellungsformen von Algorithmen



Beispiel Programmablaufplan

– Welche Aufgabe löst dieser Algorithmus?



1. Gegeben ist eine positive Zahl n
2. Speichere 1 im Ergebnis f und 1 in m
3. Multipliziere f mit m und speichere das Ergebnis in f
4. Prüfe, ob $m = n$. Wenn nicht, erhöhe m um 1 und gehe zu 3.
5. Gib f als Ergebnis

– Fakultätsberechnung (Alternative zum bisherigen Beispiel)

Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

Programmablaufpläne

- durch Pfeile verbundene grafische Symbole stellen den möglichen Ablauf eines Algorithmus dar
- Standardisiert in DIN 66001/ISO 5807
- verlieren in ihrer ursprünglichen Form ihre Praxisrelevanz, da sie zunehmend in Form von UML Aktivitätsdiagrammen Verbreitung finden

Aktivitätsdiagramm

Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

Programmablaufplan

UML-Aktivitätsdiagramm

- Unified Modeling Language (UML) als etablierter Standard in der Praxis der modernen Softwareentwicklung
- dient u.a. der Beschreibung des dynamischen Verhaltens von Softwaresystemen
- umfasst mit Aktivitätsdiagrammen auch Darstellungsform für Algorithmen
- Überarbeitung der ursprünglichen Version 1.4 in Version 2.0



Darstellungsformen von Algorithmen



Aktivitätsdiagramm

– Häufig verwendete Elemente

- Start und Ende

Start ●

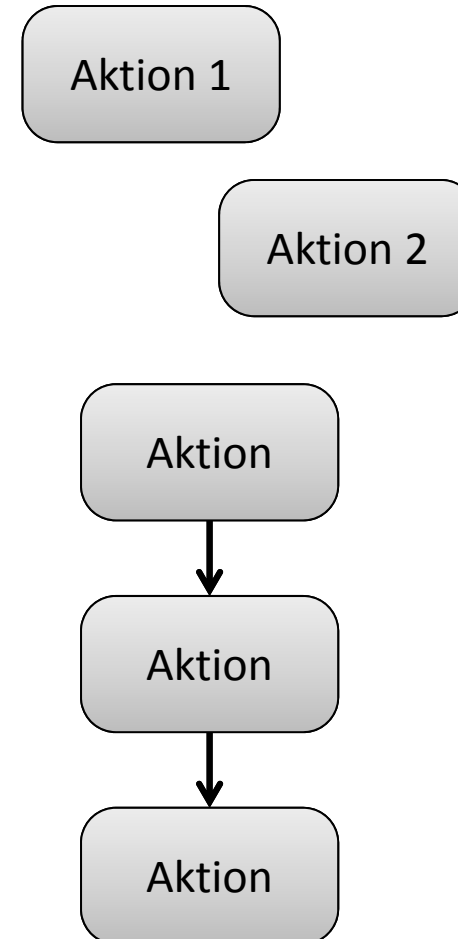
● Ende

Darstellungsformen von Algorithmen



Aktivitätsdiagramm

- Häufig verwendete Elemente
 - Start und Ende
 - Aktion und deren sequentielle Abfolge

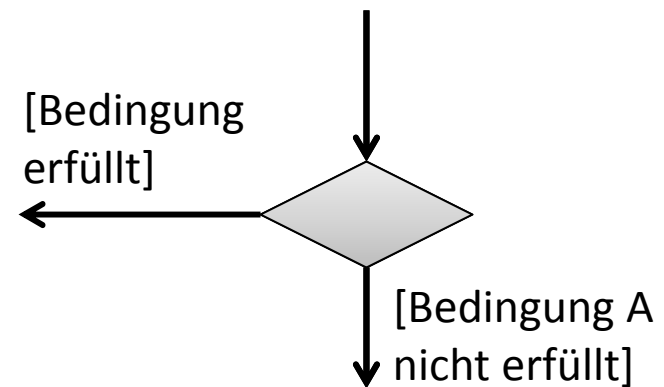


Darstellungsformen von Algorithmen



Aktivitätsdiagramm

- Häufig verwendete Elemente
 - Start und Ende
 - Aktion und deren sequentielle Abfolge
 - Bedingung/Verzweigung

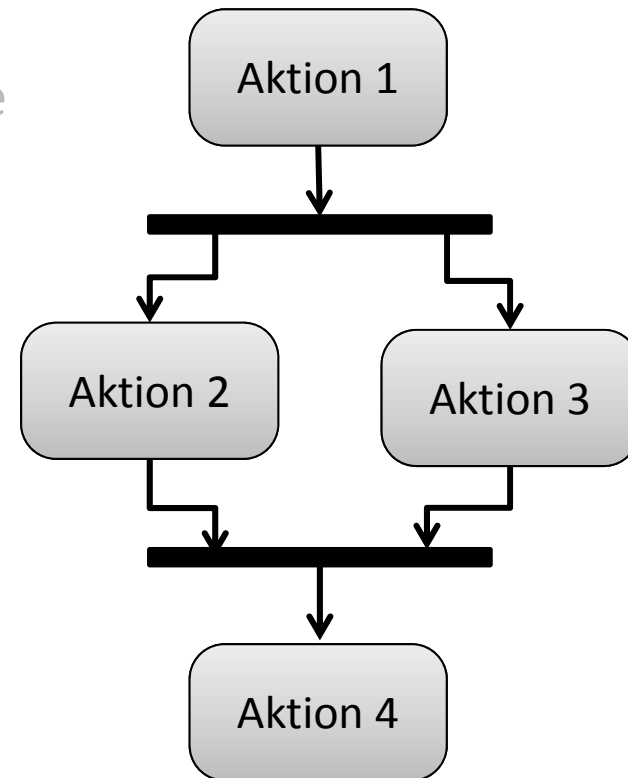


Darstellungsformen von Algorithmen



Aktivitätsdiagramm

- Häufig verwendete Elemente
 - Start und Ende
 - Aktion und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Parallele Abfolge von Aktionen

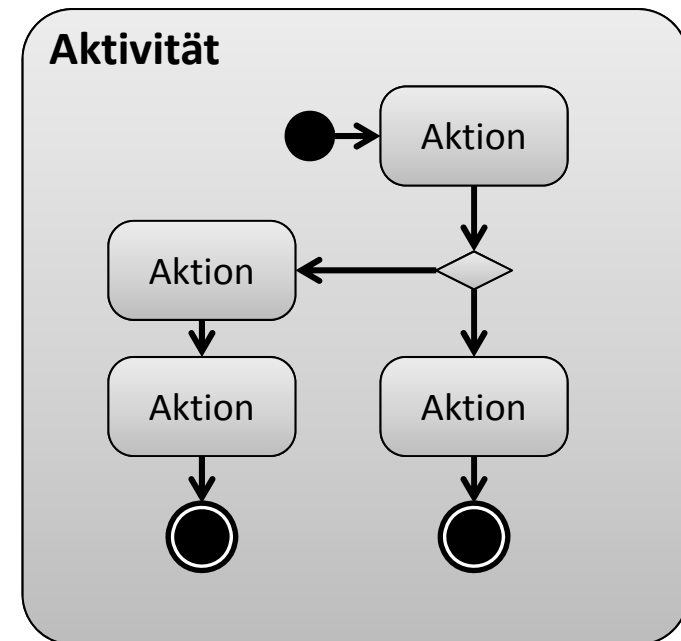


Darstellungsformen von Algorithmen



Aktivitätsdiagramm

- Häufig verwendete Elemente
 - Start und Ende
 - Aktion und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Parallele Abfolge von Aktionen
- seit UML Version 2.0
 - Elemente innerhalb einer Aktivität dargestellt

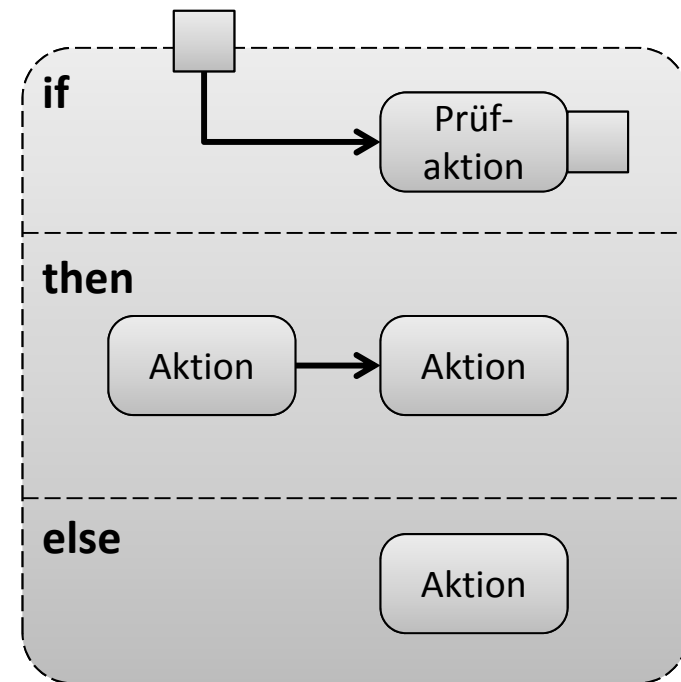


Darstellungsformen von Algorithmen



Aktivitätsdiagramm

- Häufig verwendete Elemente
 - Start und Ende
 - Aktion und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Parallele Abfolge von Aktionen
- seit UML Version 2.0
 - Elemente innerhalb einer Aktivität dargestellt
 - Verzweigung als eigenes Element



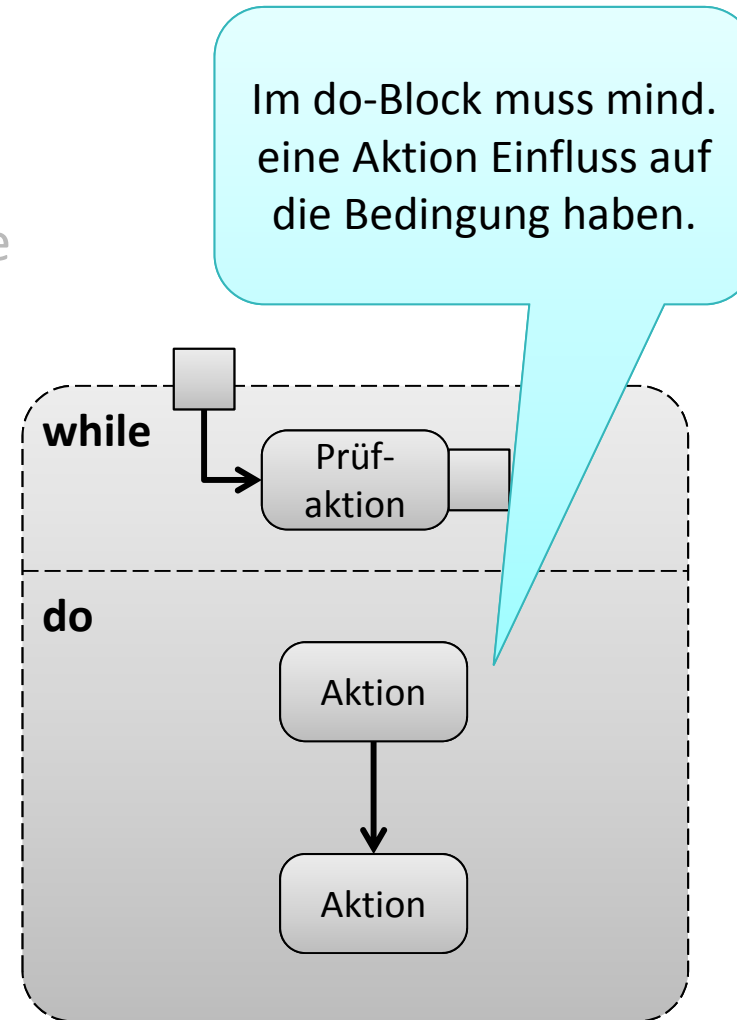
Hinweis: Beispieldarstellung für if-then-else, Alternativen möglich. (Keine Notationsvorgabe im UML-Standard)

Darstellungsformen von Algorithmen



Aktivitätsdiagramm

- Häufig verwendete Elemente
 - Start und Ende
 - Aktion und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Parallele Abfolge von Aktionen
- seit UML Version 2.0
 - Elemente innerhalb einer Aktivität dargestellt
 - Verzweigung als eigenes Element
 - Schleife als eigenes Element



Hinweis: Beispieldarstellung für while-do-Schleife, Alternativen möglich. (Keine Notationsvorgabe im UML-Standard)

Darstellungsformen von Algorithmen



Aktivitätsdiagramm

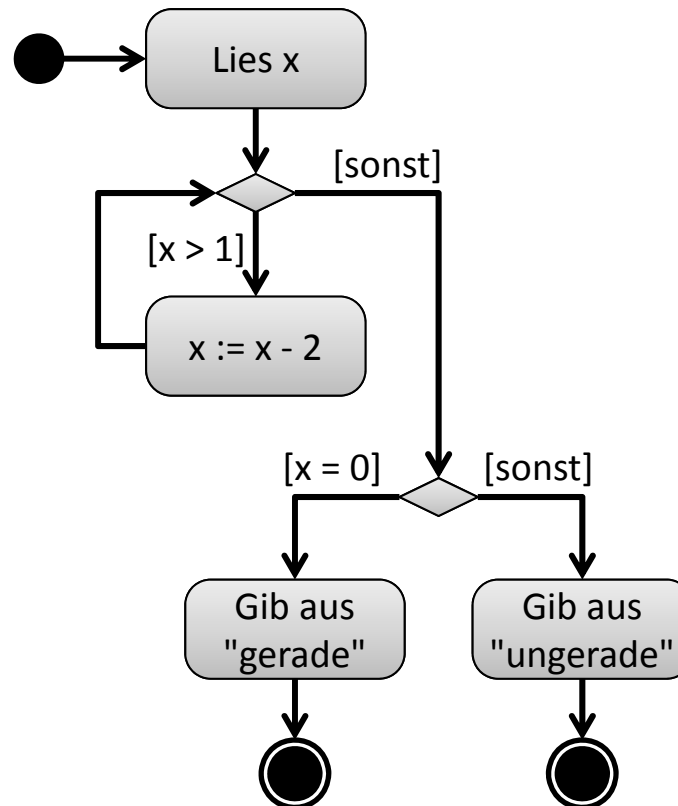
- Häufig verwendete Elemente
 - Start und Ende
 - Aktion und deren sequentielle Abfolge
 - Bedingung/Verzweigung
 - Parallele Abfolge von Aktionen
- seit UML Version 2.0
 - Elemente innerhalb einer Aktivität dargestellt
 - Verzweigung als eigenes Element
 - Schleife als eigenes Element
- weitere Elemente für komplexe Abläufe (z.B. Parameter, Schwimmbahnen)

Darstellungsformen von Algorithmen



Beispiel Aktivitätsdiagramm (UML 1.x)

– Welche Aufgabe löst dieser Algorithmus?



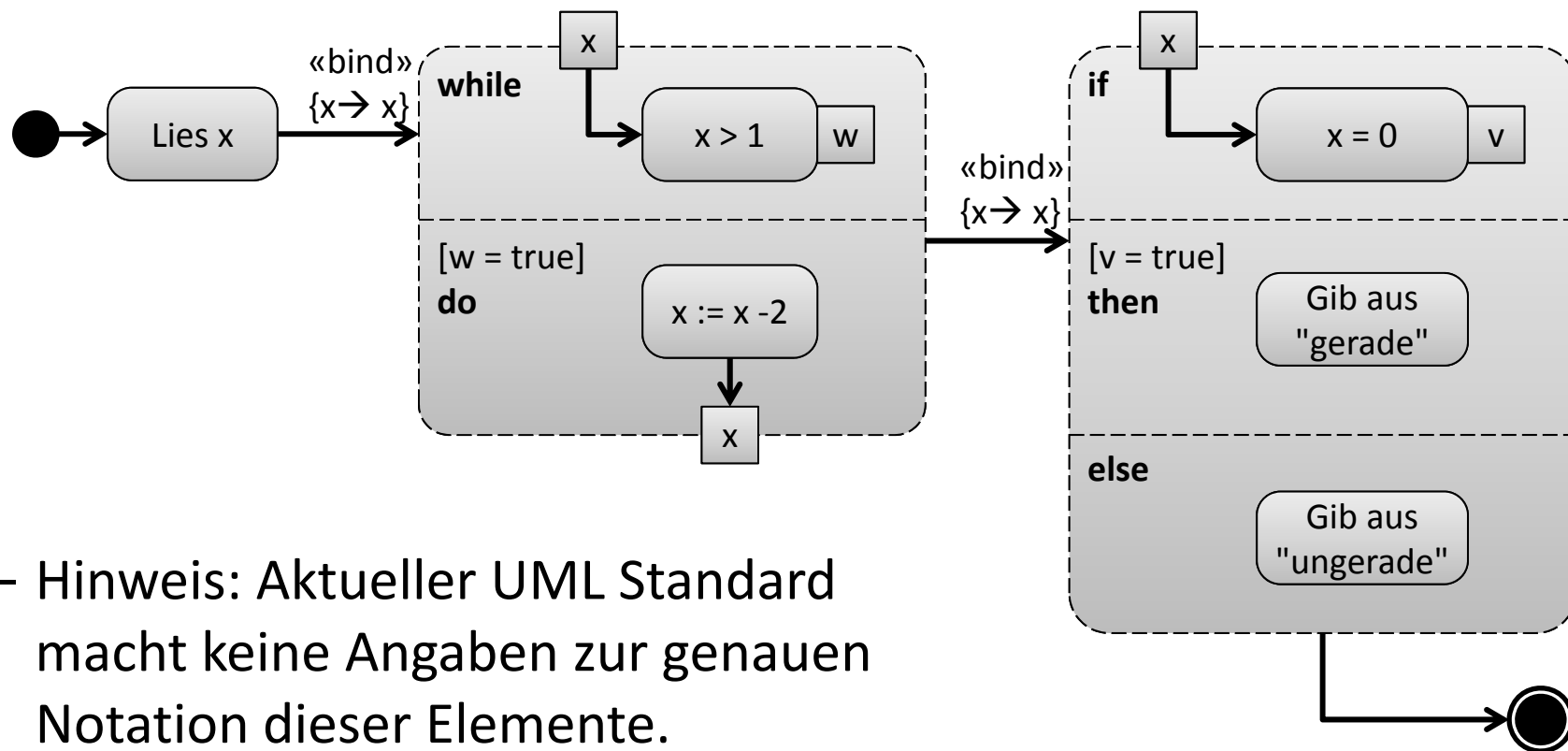
– Prüfung, ob eine eingegebene Zahl gerade ist oder ungerade.

Darstellungsformen von Algorithmen



Beispiel Aktivitätsdiagramm (UML 2.x)

- Alternative Darstellung des vorherigen Algorithmus



- Hinweis: Aktueller UML Standard macht keine Angaben zur genauen Notation dieser Elemente.

Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

Programmablaufplan

UML-Aktivitätsdiagramm

- Unified Modeling Language (UML) als etablierter Standard in der Praxis der modernen Softwareentwicklung
- dient u.a. der Beschreibung des dynamischen Verhaltens von Softwaresystemen
- umfasst mit Aktivitätsdiagrammen auch Darstellungsform für Algorithmen
- Überarbeitung der ursprünglichen Version 1.4 in Version 2.0



Darstellungsformen von Algorithmen



Pseudocode

Struktogramm (Nassi-Shneiderman-Diagramm)

Programmablaufplan

UML-Aktivitätsdiagramm

...



Bekannte Algorithmen

Für verschiedene Problemstellungen sind bereits Algorithmen entwickelt worden, die Probleme unterschiedlich lösen. Bekannt sind u.a.

- Sortieralgorithmen: Sortierung von Elementen unter Verwendung einer Ordnungsrelation (größer, kleiner, gleich)
- Suchalgorithmen: finden eines oder mehrerer Datenelemente anhand einer Eigenschaft in einer Menge von Datenelementen (Voraussetzung, dass Menge von Datenelemente sortiert ist)
- Algorithmen für Graphen: Verarbeitung von Graphen bestehend aus Knoten und Kanten (z.B. Durchlaufen eines Graphen, Kürzeste Wege finden)
- Suchalgorithmen für Texte: Suche nach Teilmustern in Texten (z.B. Boyer-Moore)
- ...

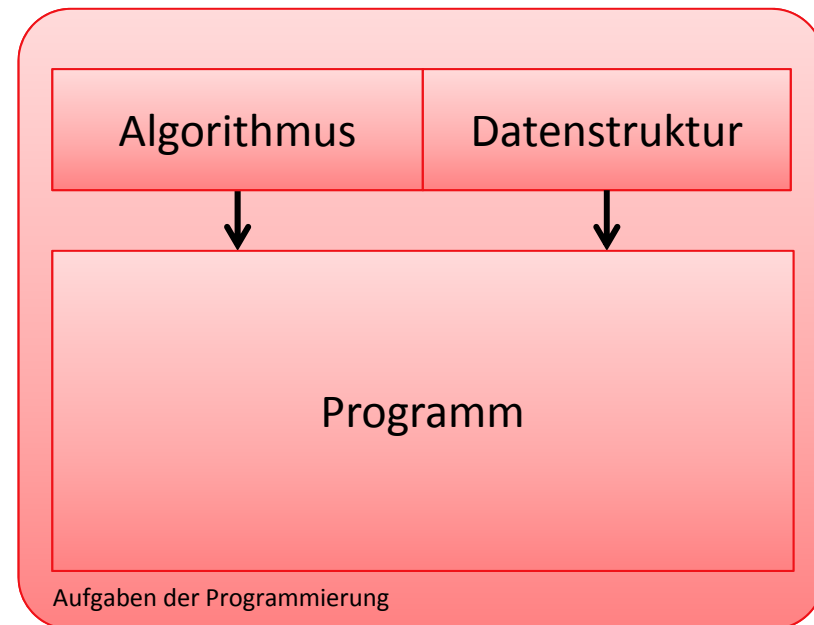
Zusammenfassung



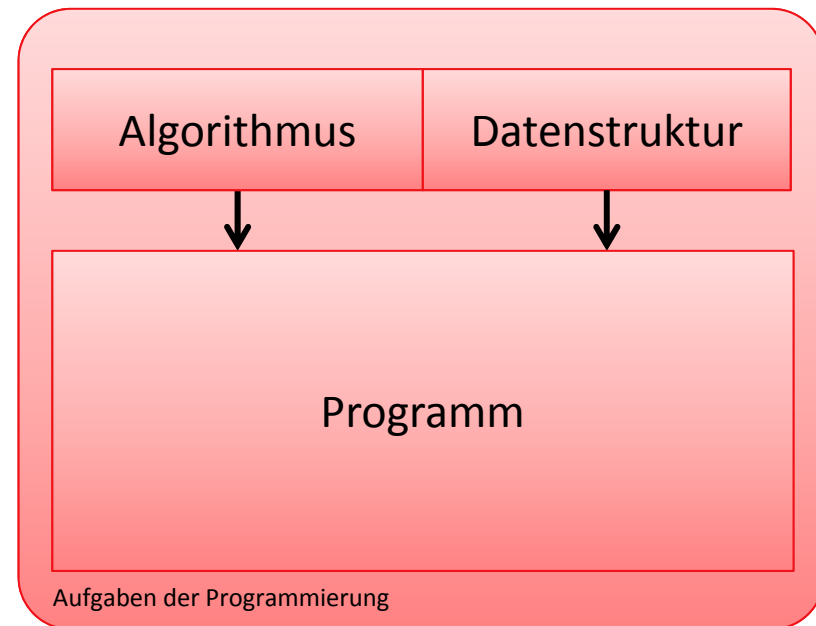
Algorithmus

- ist eine präzise und vollständig in einer eindeutigen Sprache formulierte Verarbeitungsvorschrift, die eine endliche Abfolge einzeln ausführbarer Verarbeitungsschritte vorgibt, die eine Ausgangssituation in ein Ergebnis überführen, das zur Lösung einer Aufgabenstellung/eines Problems dienen soll
- umfasst definierte Bestandteile, wie Anweisungen, Schleifen, Verzweigungen/Bedingungen, Unterprogramme
- verwendet zur präzisen Beschreibung z.B. Pseudocode, Struktogramme, Programmablaufpläne oder UML-Aktivitätsdiagramme

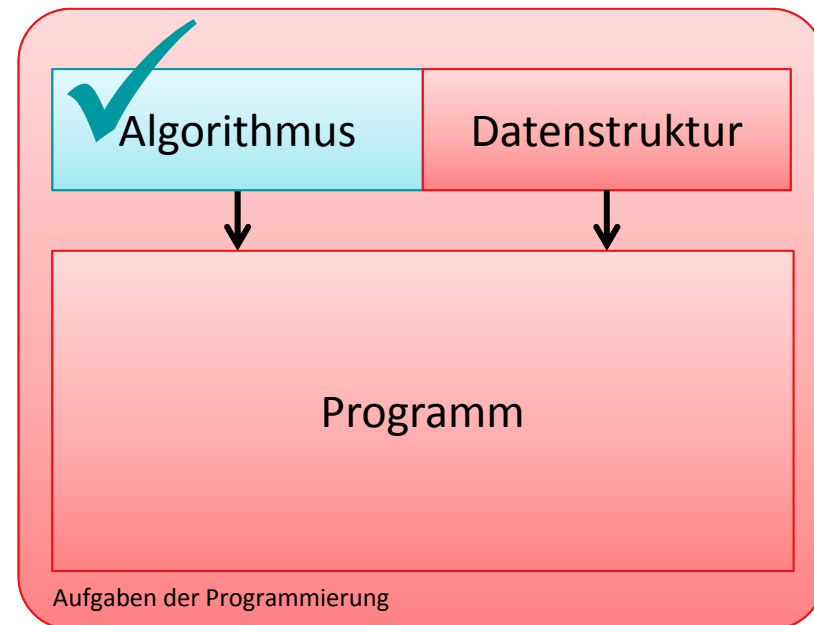
Algorithmus, Datenstruktur und Programm



Algorithmus, Datenstruktur und Programm



Algorithmus, Datenstruktur und Programm

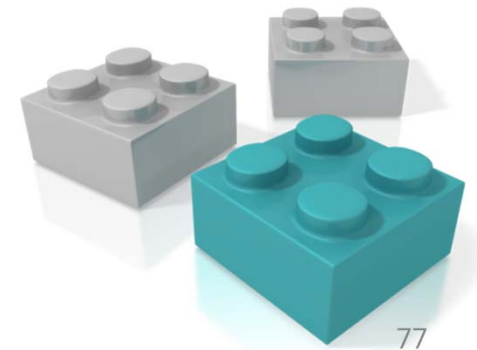


Datenelement und Datenstruktur



In einem Algorithmus zu verarbeitende Daten werden gespeichert in

- in einfachen Datenelementen
- in (komplexeren) Datenelementen/-strukturen

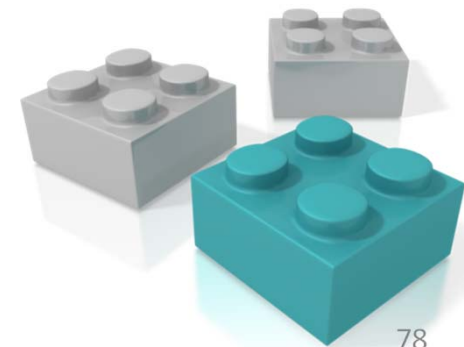


Datenelement und Datenstruktur



In einem Algorithmus zu verarbeitende Daten werden gespeichert in

- in einfachen Datenelementen
 - entsprechen einem Speicherbereich einer bestimmten Größe im Hauptspeicher
 - Wert kann zugewiesen, d.h. in dem zugehörigen Speicherbereich abgelegt werden
 - Wert kann gelesen werden, d.h. aus dem zugehörigen Speicherbereich wird der dort zu findende Wert gelesen
- in (komplexeren) Datenelementen/-strukturen



Einfache Datenelemente



Beispiele

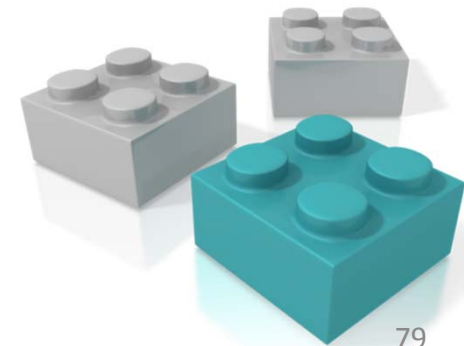
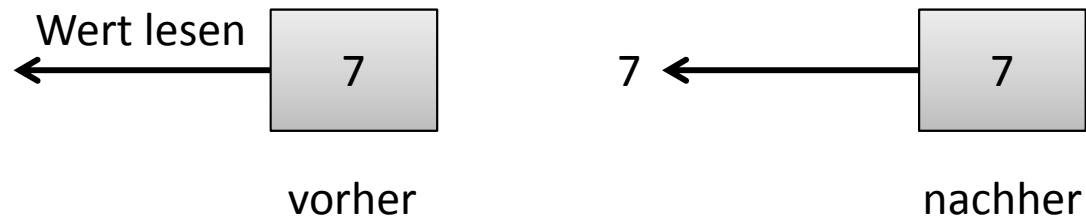
- einfaches Datenelement ohne initialen Wert



- Schreibender Zugriff auf einfaches Datenelement



- Lesender Zugriff auf einfaches Datenelement

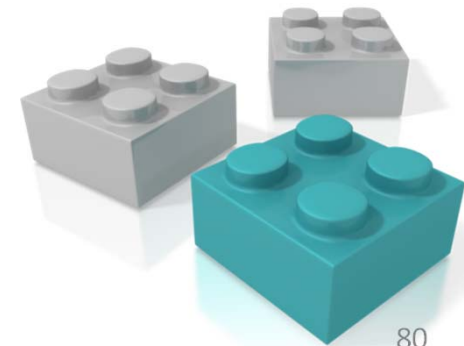


Datenelement und Datenstruktur



In einem Algorithmus zu verarbeitende Daten werden gespeichert in

- in einfachen Datenelementen
 - entsprechen einem Speicherbereich einer bestimmten Größe im Hauptspeicher
 - Wert kann zugewiesen, d.h. in dem zugehörigen Speicherbereich abgelegt werden
 - Wert kann gelesen werden, d.h. aus dem zugehörigen Speicherbereich wird der dort zu findende Wert gelesen
- in (komplexeren) Datenelementen/-strukturen

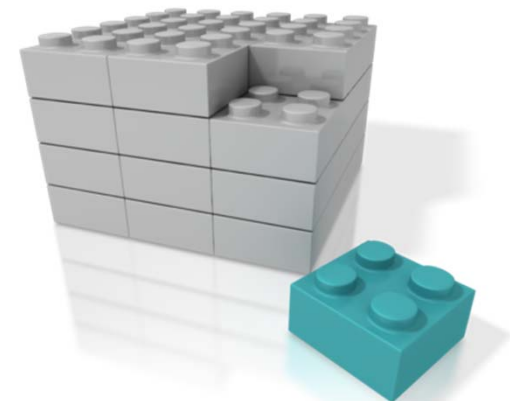


Datenelement und Datenstruktur



In einem Algorithmus zu verarbeitende Daten werden gespeichert in

- einfaches Datenelement
- in (komplexeren) Datenelementen/-strukturen
 - umfassen in der Regel mehrere einfache Datenelemente
 - organisieren die Datenelemente in einer bestimmten Form
 - ermöglichen den lesenden und schreibenden Zugriff auf Elemente innerhalb der Datenstruktur
 - weitere Aktionsmöglichkeiten für Zugriff und Verwaltung der Datenstruktur und ihrer Elemente, u.a.
 - Einfügen und Entfernen
 - Suchen
 - Ändern
 - Iteration über alle Werte in der Datenstruktur

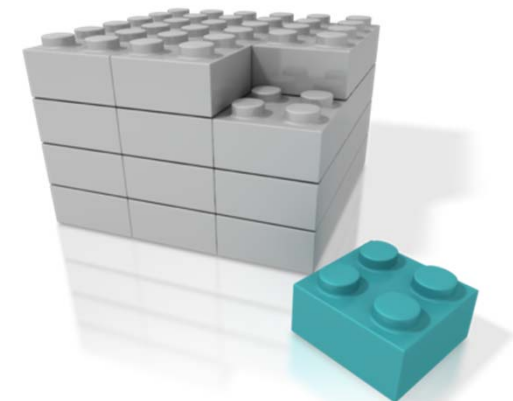
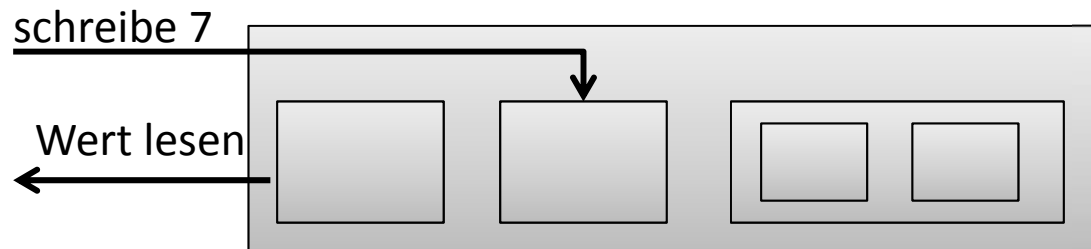




Komplexe Datenelemente

Komplexe Datenelemente

- Aufbau
 - fassen mehrere einfache Datenelemente oder andere komplexe Datenelemente zusammen
 - Repräsentieren häufig Dinge und Konzepte der Realität
- Beispiel: komplexes Datenelement "Person" fasst einfache Datenelemente "Name", "Vorname" und komplexes Datenelement "Adresse" zusammen
- Zugriff auf einzelne Elemente des komplexen Datenelementes möglich (Lesen, Schreiben)





Datenstrukturen

Sequentielle Liste/Feld (Array)

Verkettete Liste

Map

Stapel (Stack)

Schlange (Queue)

Graph

Baum

...

Datenstrukturen



Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich

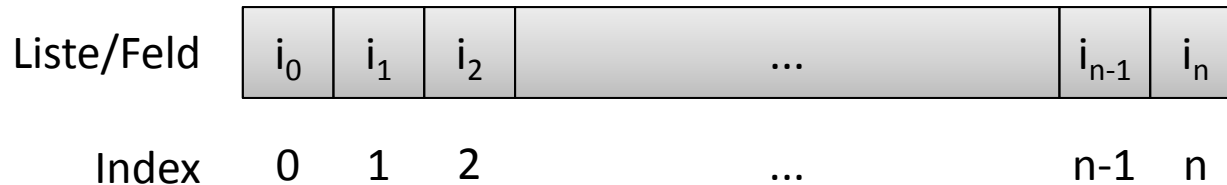


Datenstrukturen



Sequentielle Liste/Feld (Array)

- Lesender Zugriff über Index frei möglich



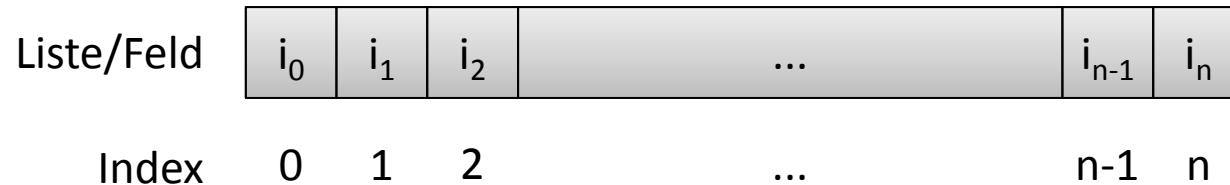
- Einfügen



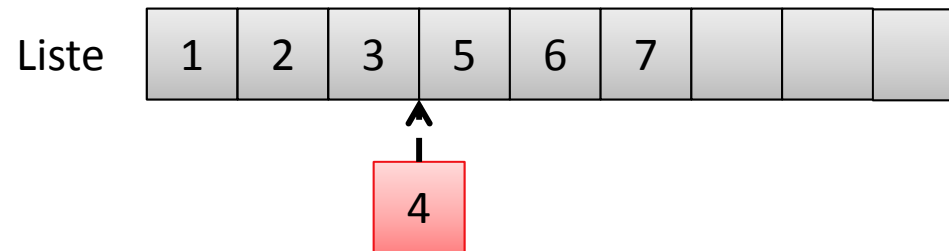
Datenstrukturen

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich



– Einfügen

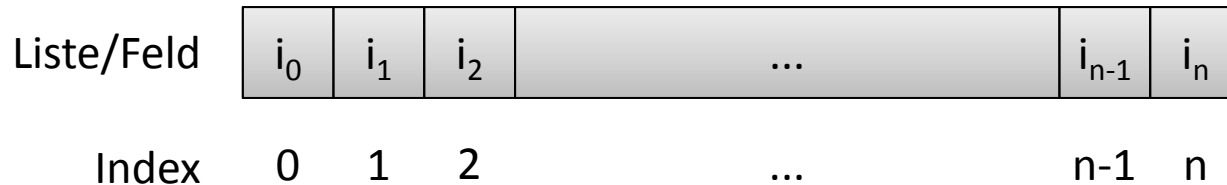




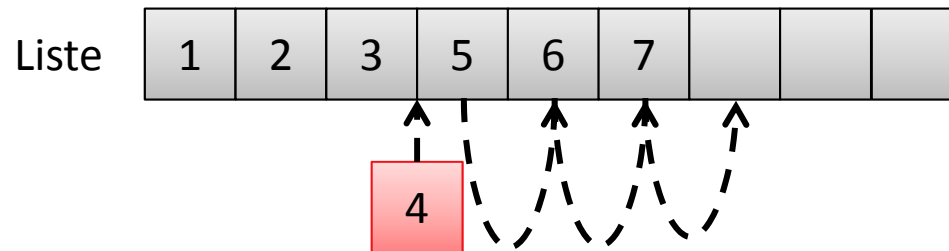
Datenstrukturen

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich



– Einfügen

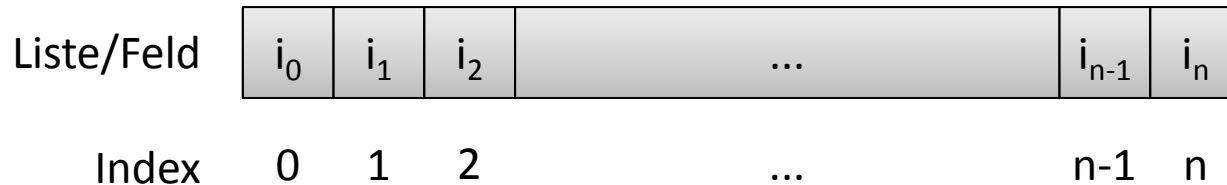




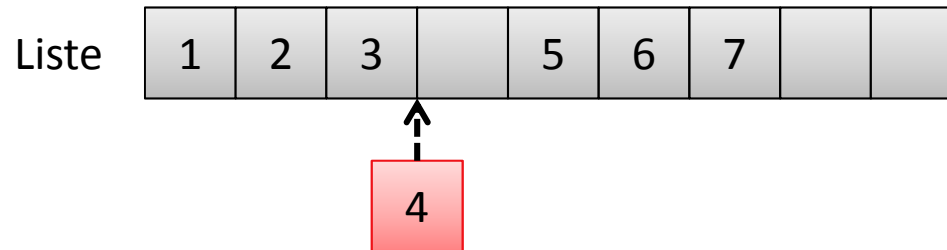
Datenstrukturen

Sequentielle Liste/Feld (Array)

- Lesender Zugriff über Index frei möglich



- Einfügen

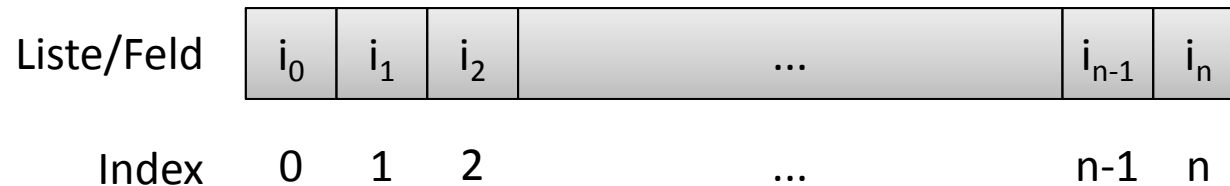




Datenstrukturen

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich



– Einfügen

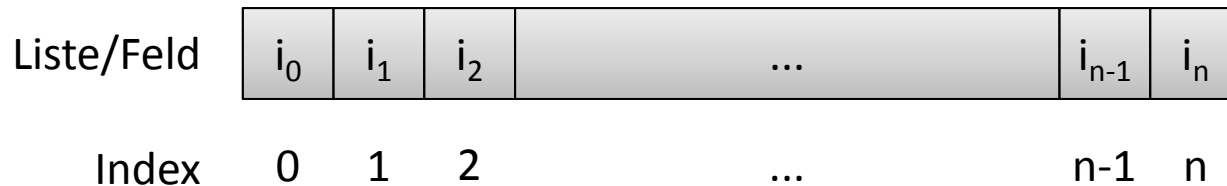


Datenstrukturen



Sequentielle Liste/Feld (Array)

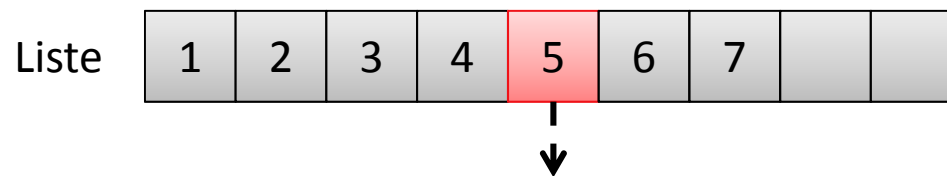
– Lesender Zugriff über Index frei möglich



– Einfügen



– Entfernen

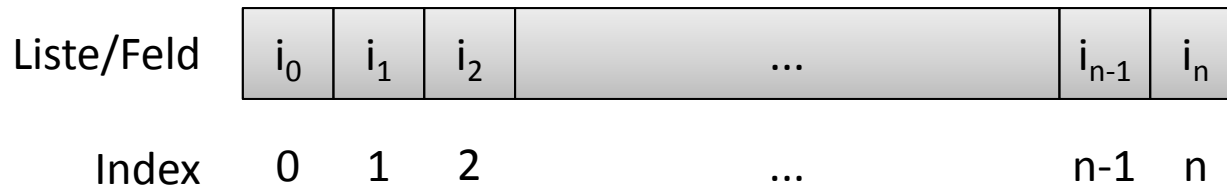


Datenstrukturen



Sequentielle Liste/Feld (Array)

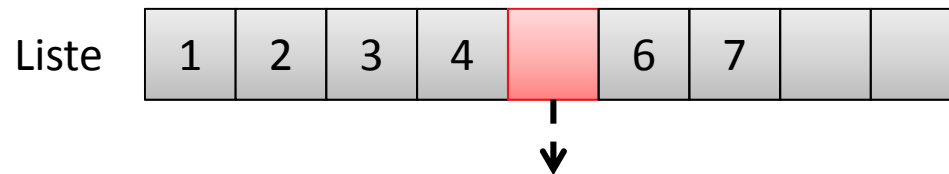
– Lesender Zugriff über Index frei möglich



– Einfügen



– Entfernen





Datenstrukturen

Sequentielle Liste/Feld (Array)

– Lesender Zugriff über Index frei möglich



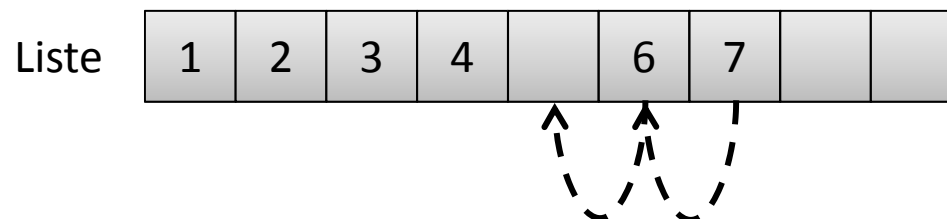
– Einfügen

Index

0	1	2	...	n-1	n
---	---	---	-----	-----	---



– Entfernen



Datenstrukturen



Sequentielle Liste/Feld (Array)

– Suchen: Zugriff über Index frei möglich

Liste/Feld	i_0	i_1	i_2	...	i_{n-1}	i_n
------------	-------	-------	-------	-----	-----------	-------

– Einfügen

Index	0	1	2	...	n-1	n
-------	---	---	---	-----	-----	---

Liste	1	2	3	4	5	6	7		
-------	---	---	---	---	---	---	---	--	--

– Entfernen

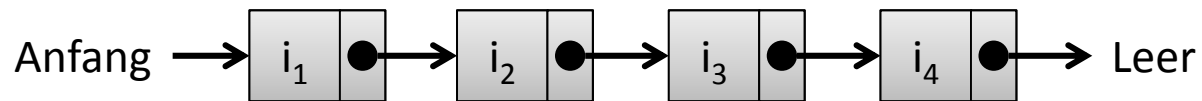
Liste	1	2	3	4	6	7			
-------	---	---	---	---	---	---	--	--	--

Datenstrukturen



Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste

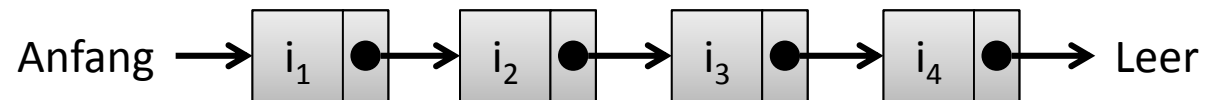


Datenstrukturen



Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste



- Suchen: Liste entlang der Verweise durchlaufen

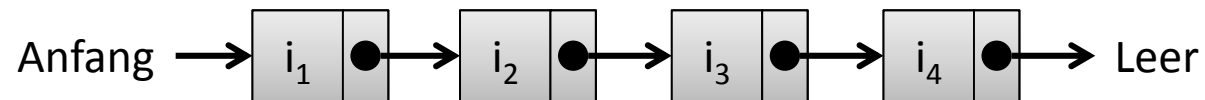


Datenstrukturen

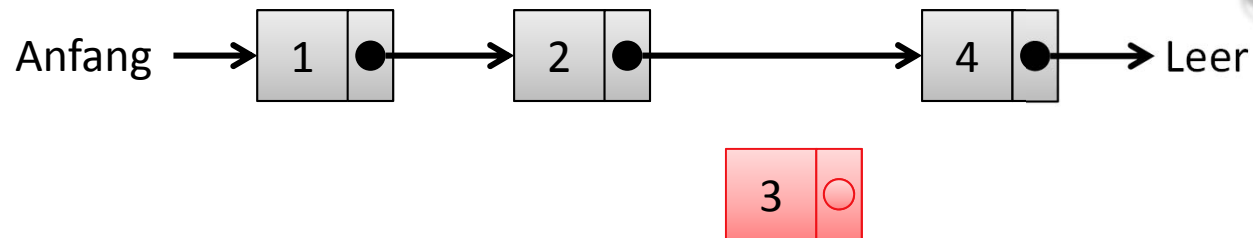


Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste



- Suchen: Liste entlang der Verweise durchlaufen
- Einfügen:

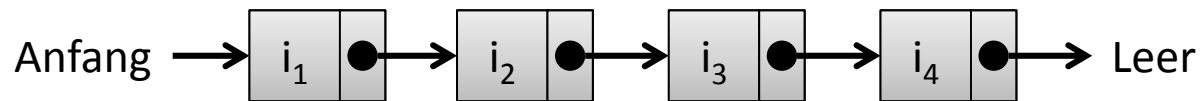


Datenstrukturen

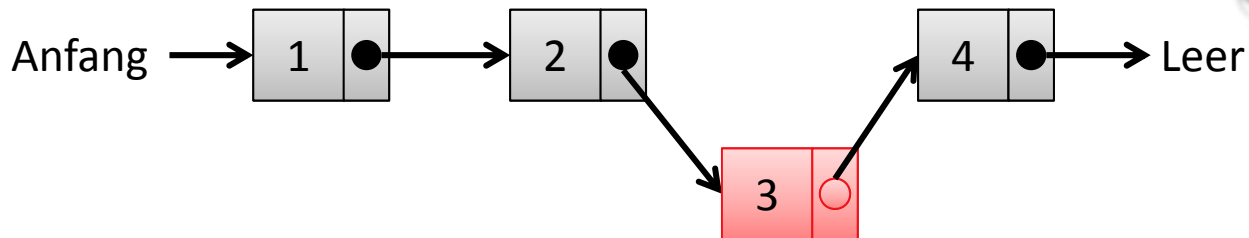


Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste



- Suchen: Liste entlang der Verweise durchlaufen
- Einfügen: Verweise "umhängen"

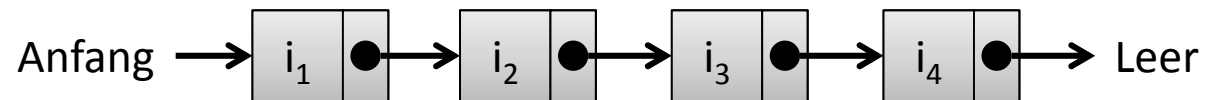


Datenstrukturen

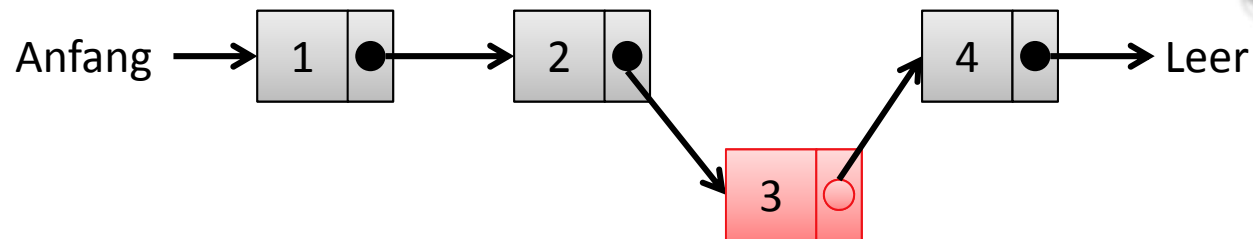


Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste



- Suchen: Liste entlang der Verweise durchlaufen
- Einfügen: Verweise "umhängen"
- Löschen:

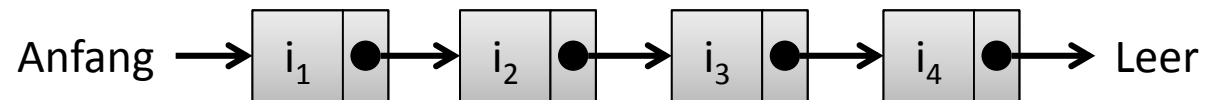


Datenstrukturen

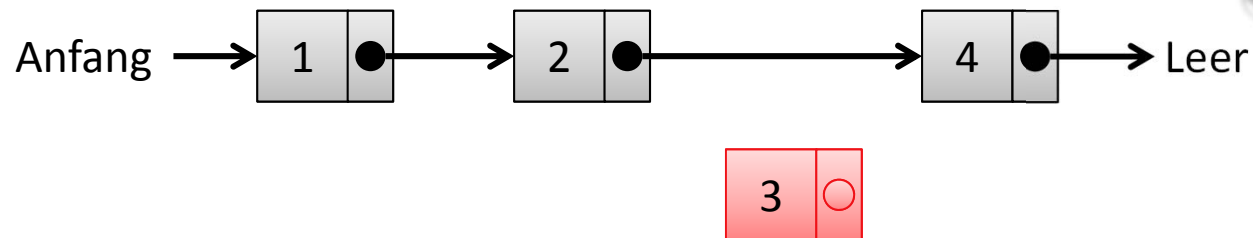


Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste



- Suchen: Liste entlang der Verweise durchlaufen
- Einfügen: Verweise "umhängen"
- Löschen: Verweise "umhängen"

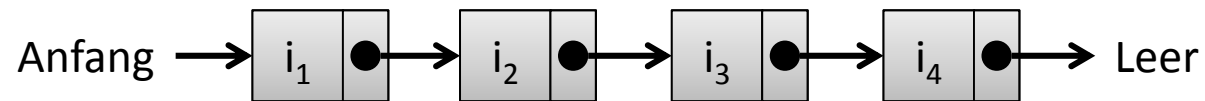


Datenstrukturen

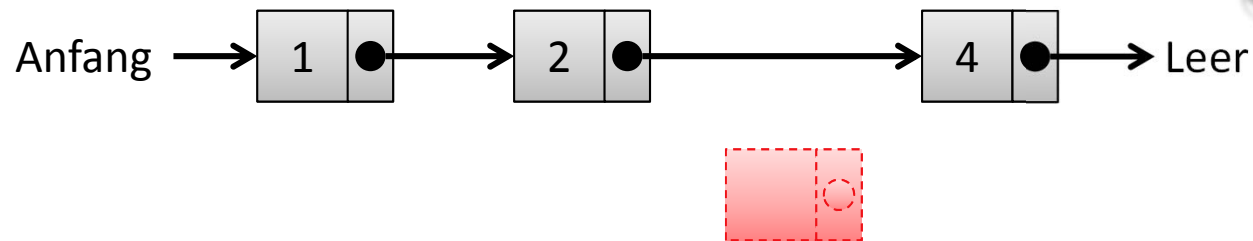


Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste



- Suchen: Liste entlang der Verweise durchlaufen
- Einfügen: Verweise "umhängen"
- Löschen: Verweise "umhängen"

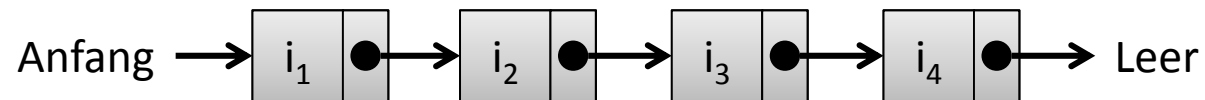


Datenstrukturen



Verkettete Liste

- Aufbau: jedes Listenelement hat einen Wert und einen Verweis auf seinen Nachfolger, das letzte Element verweist ins "Leere" und es gibt einen Verweis auf den Anfang der Liste



- Suchen: Liste entlang der Verweise durchlaufen
- Einfügen: Verweise "umhängen"
- Löschen: Verweise "umhängen"
- Sonderfälle: am Anfang/Ende einfügen/löschen

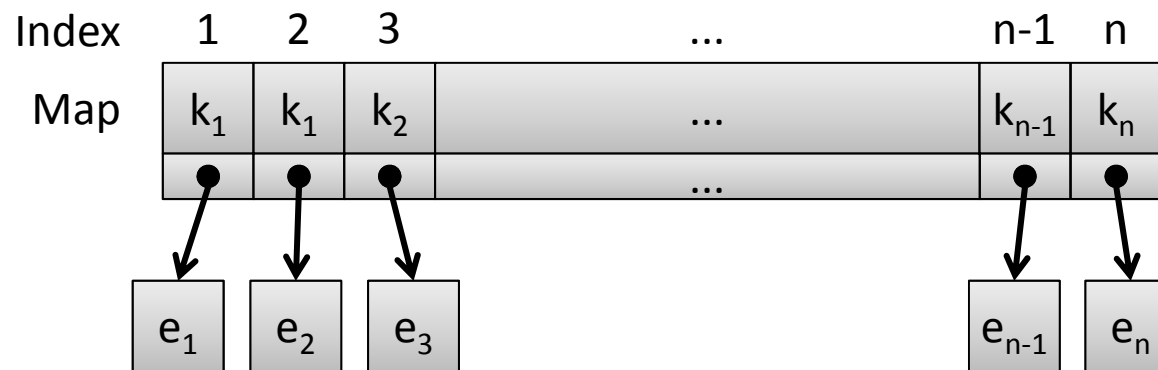


Datenstrukturen

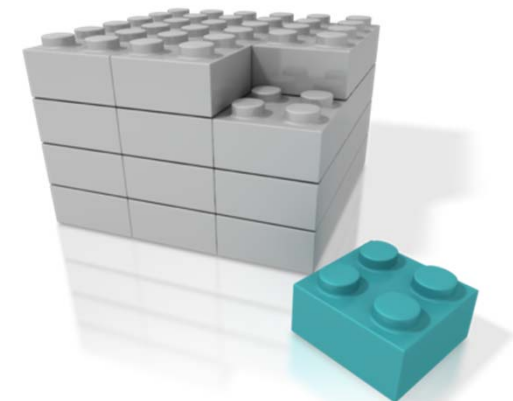


Map

- Lesender Zugriff
 - über Schlüssel (Key) frei möglich,
 - teilweise zusätzlich Zugriff über einen Index möglich (z.B. in VBA)



- Einfügen
 - mit Angabe eines Schlüssels (der in VBA noch nicht vergeben sein darf)
 - Vergrößerung erfolgt bei Bedarf automatisch
- Entfernen über Schlüssel bzw. Index

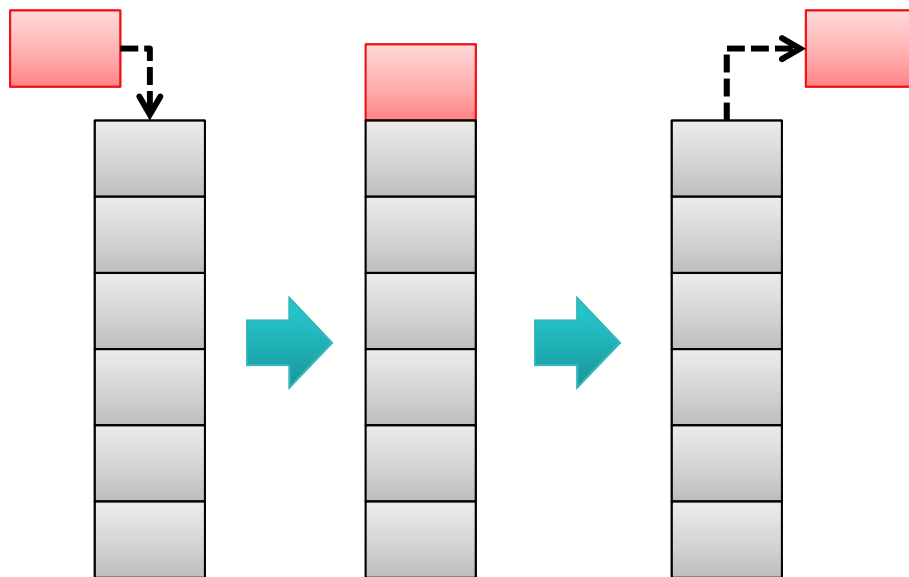


Datenstrukturen



Stapel (Stack)

- Zugriff nach dem LIFO-Prinzip (last-in first-out)
- Einfügen nur am Anfang der Liste (Push)
- Entfernen nur am Anfang der Liste (Pop)

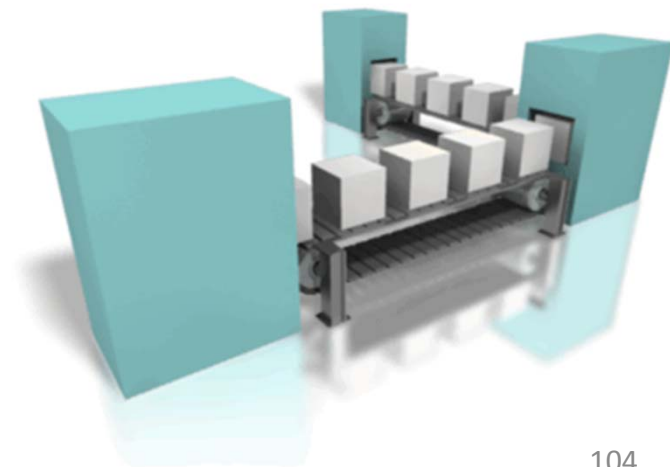
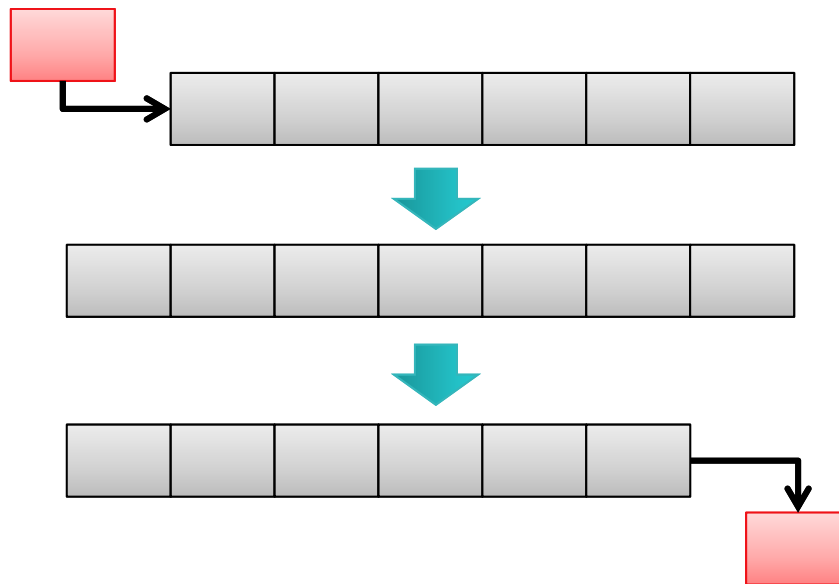


Datenstrukturen



Schlange (Queue)

- Einfügen nur am Anfang und Entnehmen nur am Ende
- FIFO-Prinzip (first-in-first-out)
- kein direkter Zugriff auf Element an bestimmter Position
- Beispiele: Warteschlange im Supermarkt, Druckerjobs



Datenstrukturen



Graph

- bestehend aus Knoten, die die Datenelemente repräsentieren und Kanten, die Verweise auf andere Datenelemente darstellen
- ähnlich der verketteten Liste, jedoch mit prinzipiell beliebig vielen Verweisen auf andere Elemente (Knoten)
- zyklischer Graph enthält Verweise auf Knoten, die Vorgänger des verweisenden Knoten sind
- Suchen: Durchlaufen der Knoten im Graphen
- Einfügen, Löschen: Umhängen von Verweisen (Kanten)

Datenstrukturen



Baum

- Spezialform des Graphen
- besteht aus Datenelementen, die Verweise auf andere Datenelemente umfassen, wobei Zyklen nicht zulässig sind
- Bäume, bei denen Datenelemente entsprechend ihrer Werte sortiert sind und sie max. 2 Verweise haben, sind von besonderer Bedeutung (Binärbaum) für effiziente Suchalgorithmen
- Suchen: Durchlaufen der Knoten im Graphen
- Einfügen, Löschen: Umhängen von Verweisen (Kanten)



Komplexe Datenelemente und -strukturen

Sequentielle Liste/Feld (Array)

Verkettete Liste

Map

Stapel (Stack)

Schlange (Queue)

Graph

Baum

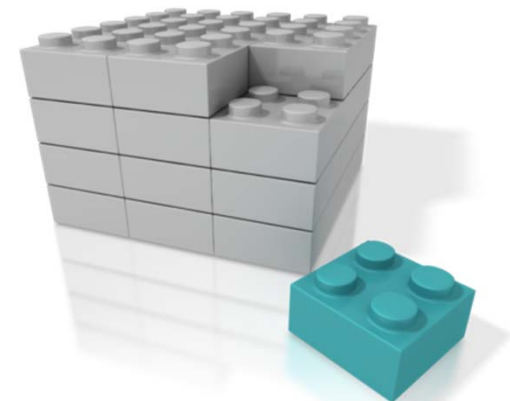
...

Datenelement und Datenstruktur



In einem Algorithmus zu verarbeitende Daten werden gespeichert in

- einfaches Datenelement
- in (komplexeren) Datenelementen/-strukturen
 - umfassen in der Regel mehrere einfache Datenelemente
 - organisieren die Datenelemente in einer bestimmten Form
 - ermöglichen den lesenden und schreibenden Zugriff auf Elemente innerhalb der Datenstruktur
 - weitere Aktionsmöglichkeiten für Zugriff und Verwaltung der Datenstruktur und ihrer Elemente, u.a.
 - Einfügen und Entfernen
 - Suchen
 - Ändern
 - Iteration über alle Werte in der Datenstruktur



Datenelement und Datenstruktur



In einem Algorithmus zu verarbeitende Daten werden gespeichert in

- in einfachen Datenelementen
- in (komplexeren) Datenelementen/-strukturen

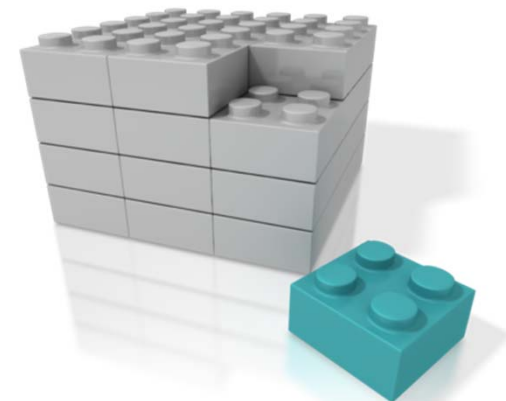


Datenelement/-struktur und Algorithmus

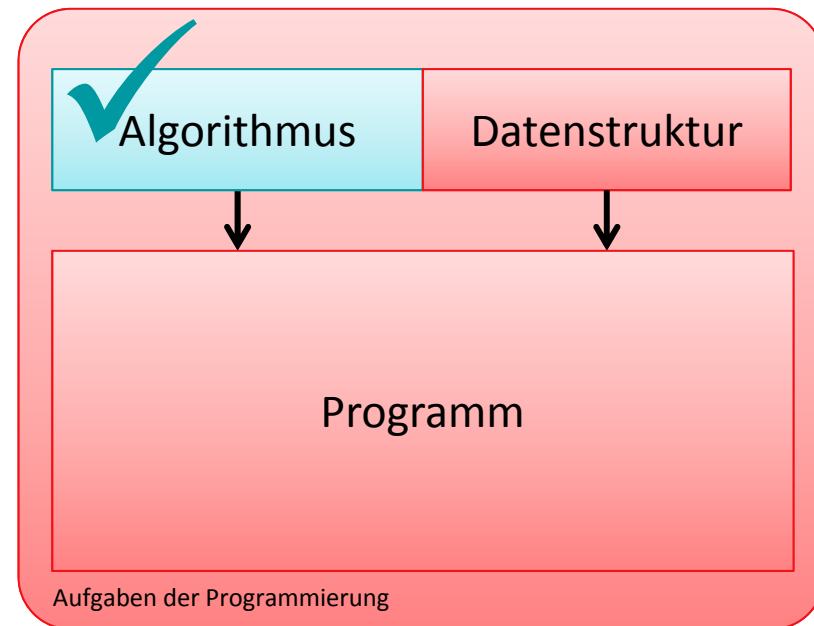


In Algorithmen

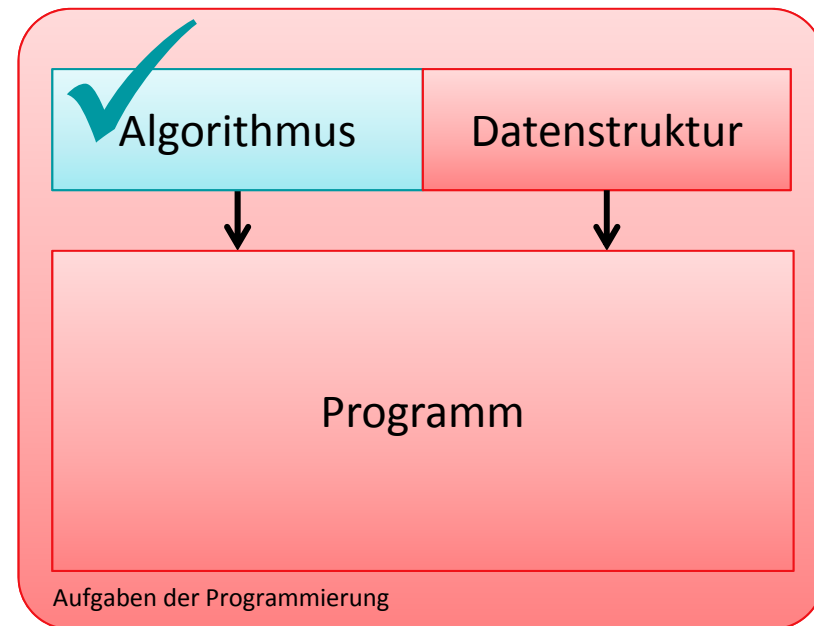
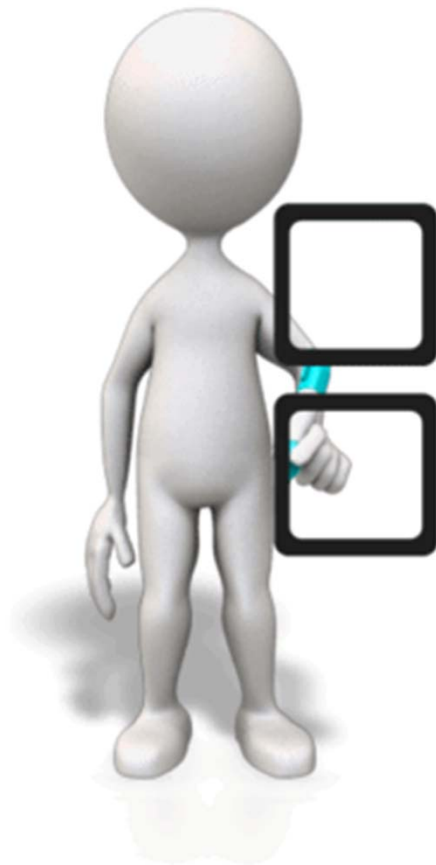
- könnten verschiedene Datenelemente und Datenstrukturen verwendet werden
- deshalb die verwendeten Datenelemente und Datenstrukturen bekannt gemacht werden
- man spricht in diesem Zusammenhang von "Deklaration"



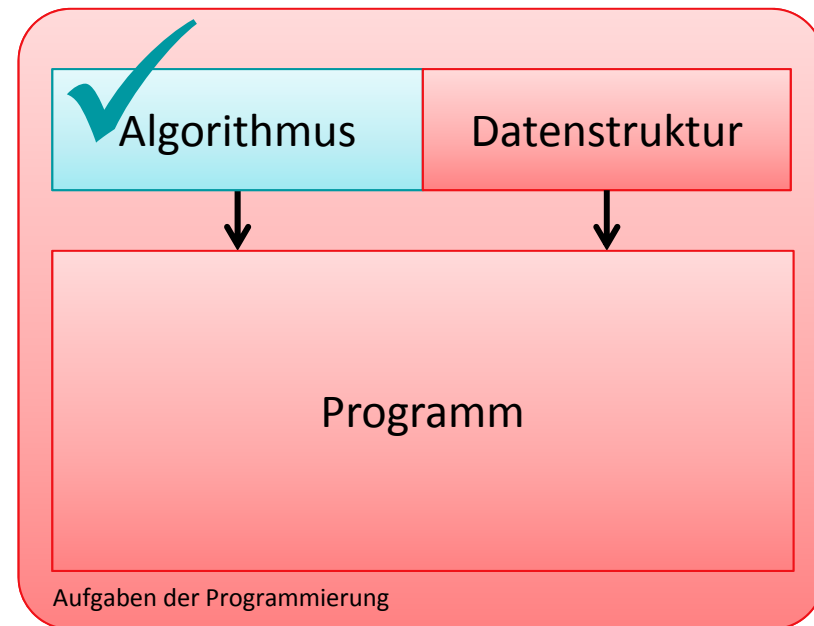
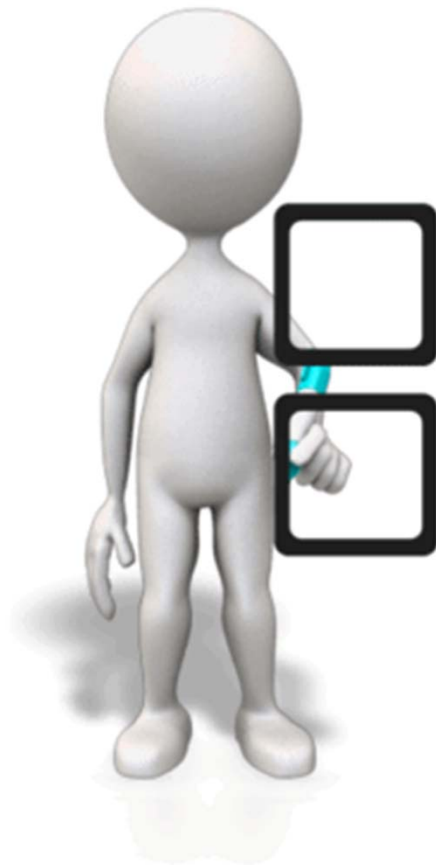
Algorithmus, Datenstruktur und Programm



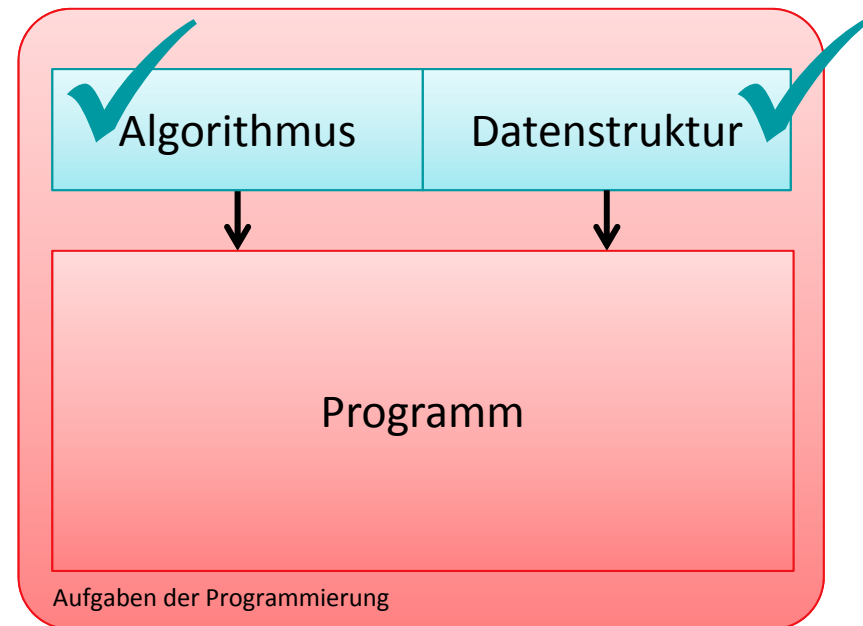
Algorithmus, Datenstruktur und Programm



Algorithmus, Datenstruktur und Programm



Algorithmus, Datenstruktur und Programm



Programm



Definition Programm

- mit den Sprachmitteln einer konkreten Programmiersprache ausgedrückter Algorithmus in Verbindung mit den ebenso ausgedrückten Datenstrukturen
- dient der Ausführung in einem Computer

Programmiersprache/Maschinensprache



Programmiersprache

- formale Sprache zur Formulierung von Programmen
- besitzt eine Grammatik, die die Regeln für den Aufbau und die Bedeutung der Sprache festlegt
 - präzise festgelegte Syntax
 - eindeutige und widerspruchsfreie Semantik

Maschinensprache

- Programmiersprache, in der alle Arbeitsschritte binär (d.h. als 0 oder 1) ausgedrückt werden
- Programm in Maschinensprache ist unmittelbar abarbeitungsfähig
- jeder Prozessortyp besitzt eigene Maschinensprache, da die Bedeutung einer Sequenz aus 0 und 1 von Hardware abhängig



Programmcode/Maschinencode

Definition Programmcode/Quellcode

- Darstellung des Algorithmus in der Programmiersprache
- kann nicht direkt ausgeführt werden, ist (vergleichsweise) gut zu lesen und verständlich

Definition Maschinencode

- tatsächlich ausführbarer Programmcode bestehend aus Befehlen des Befehlsvorrates eines Computers
- wird aus dem Quellcode eines Programms erzeugt
- ist in Maschinensprache formuliert



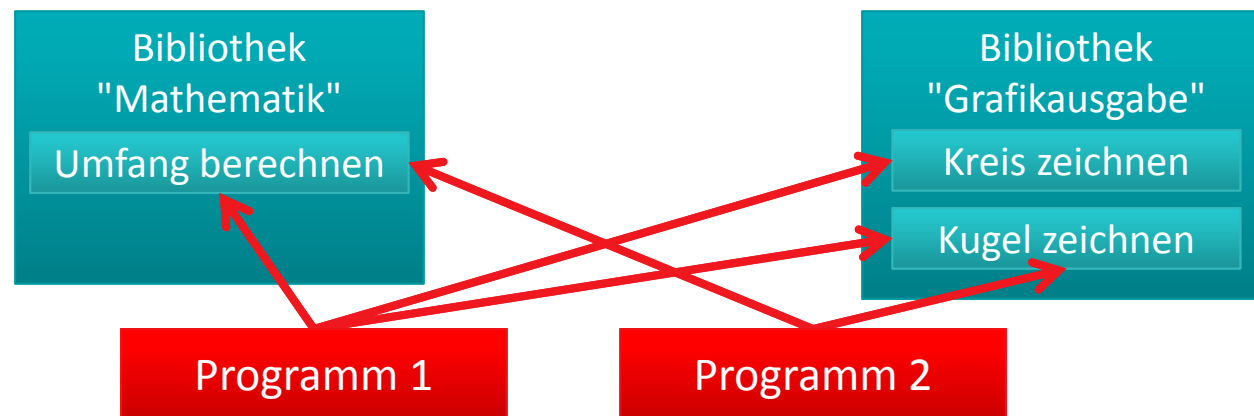
Bibliothek

Um Programme nicht stets komplett neu zu schreiben, können wiederverwendbare Teile zusammengefasst werden. Eine Möglichkeit hierfür sind Bibliotheken.

Definition Bibliothek

- Zusammenfassung von Programmteilen, die in anderen Programmen eingebunden und dadurch wiederverwendet werden können
- dient meist der Lösung einer abgegrenzten Funktionalität

Beispiel



Übersetzung von Quellcode in Maschinencode



Wie wird der Programmcode der Programmiersprache in Maschinencode überführt?

- Compiler
- Interpreter

Übersetzung von Quellcode in Maschinencode



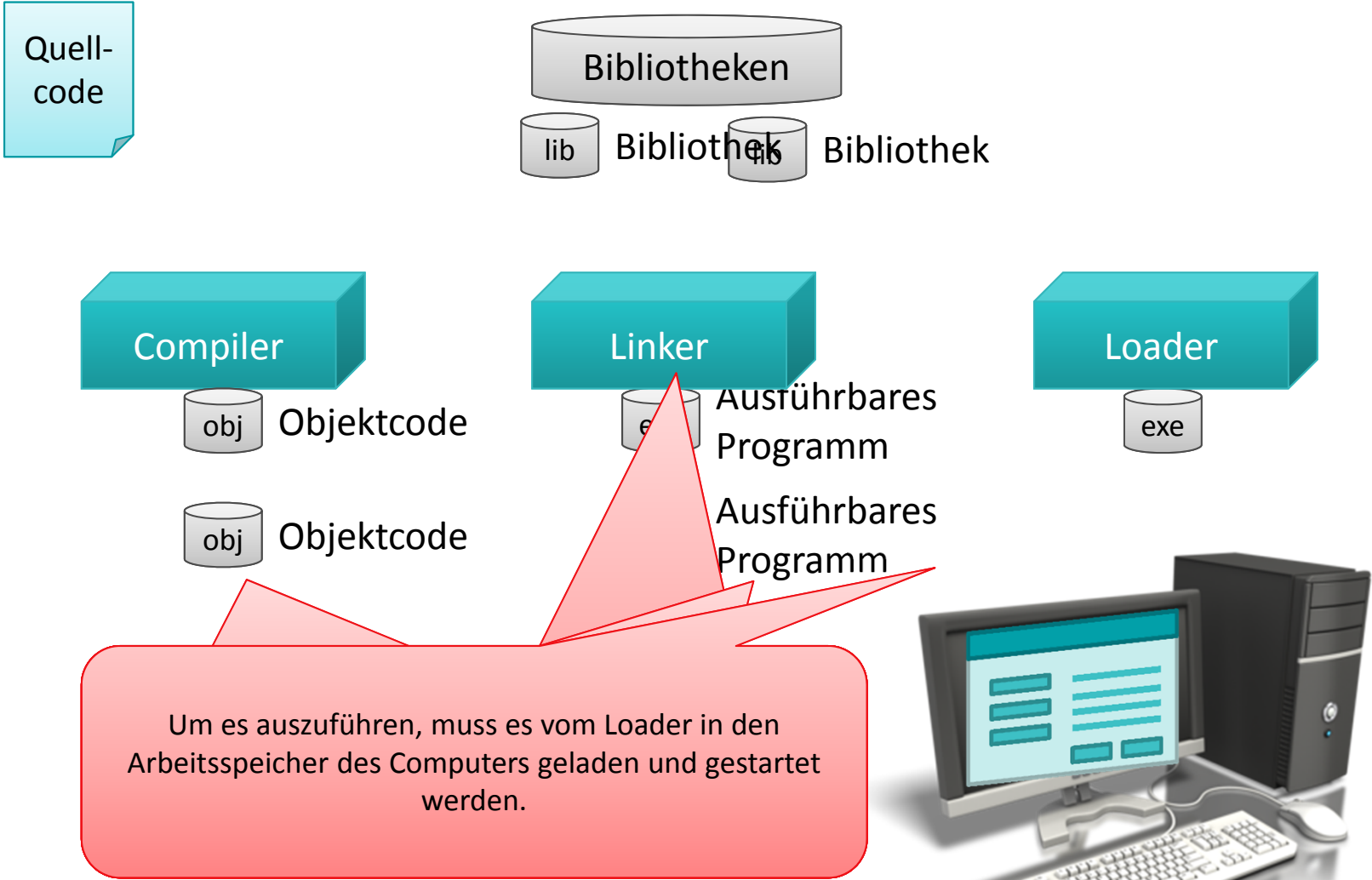
Wie wird der Programmcode der Programmiersprache in Maschinencode überführt?

– Compiler

- fertiger Programmcode wird vollständig dem **Compiler** übergeben
- Übersetzung in Zwischenformat (Objektcode)
- Zusammenführung benötigten Zusatzfunktionen (Bibliotheken) durch **Linker** erzeugt ausführbares Programm
- ausführbares Programm wird durch **Loader** in den Arbeitsspeicher geladen und ausgeführt

– Interpreter

Compiler



Quelle: [7]

Übersetzung von Quellcode in Maschinencode



Wie wird der Programmcode der Programmiersprache in Maschinencode überführt?

- Compiler
- Interpreter

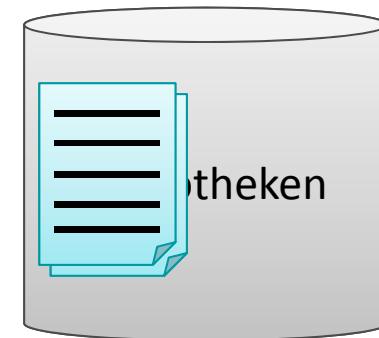
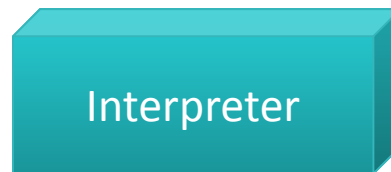
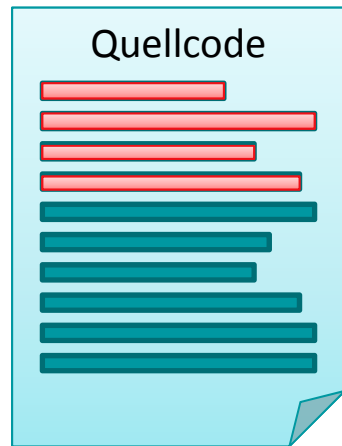
Übersetzung von Quellcode in Maschinencode



Wie wird der Programmcode der Programmiersprache in Maschinencode überführt?

- Compiler
- Interpreter
 - liest eine Anweisung aus dem Quellcode
 - übersetzt sie in Maschinencode
 - werden von der Anweisung Zusatzfunktionen (Bibliotheken) benötigt, werden diese hinzugefügt
 - Anweisung wird ausgeführt
 - mit der nächsten Anweisung wird fortgefahren

Interpreter



Weitere Anweisungen werden entsprechend jeweils Anweisung für Anweisung verarbeitet.



Dann wird die nächste Anweisung gelesen. Ist zu ihrer Ausführung eine Bibliotheksfunktion notwendig, wird zusätzlicher Code aus der Bibliothek angefordert und eingebunden. Der erzeugte Maschinencode wird geladen und ausgeführt.

Quelle: [7]

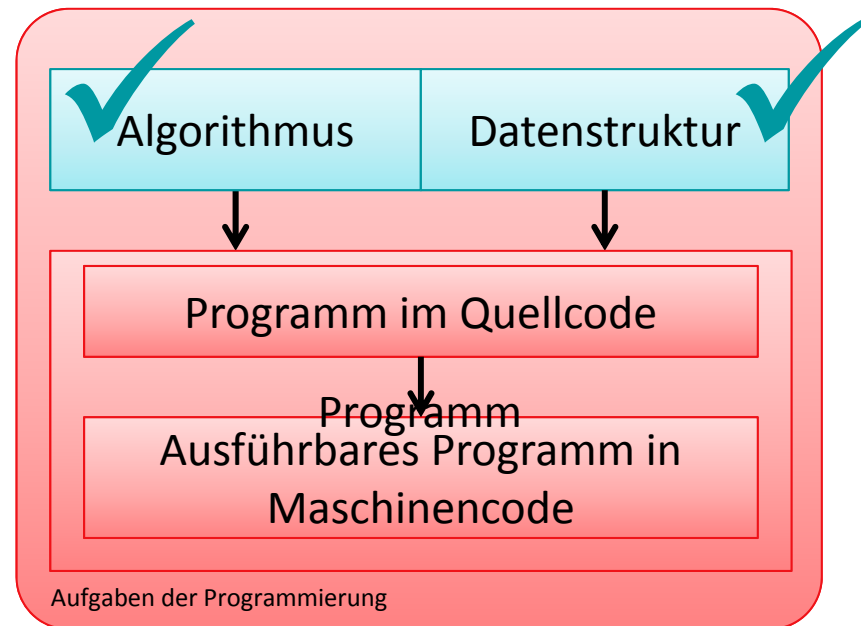
Übersetzung von Quellcode in Maschinencode



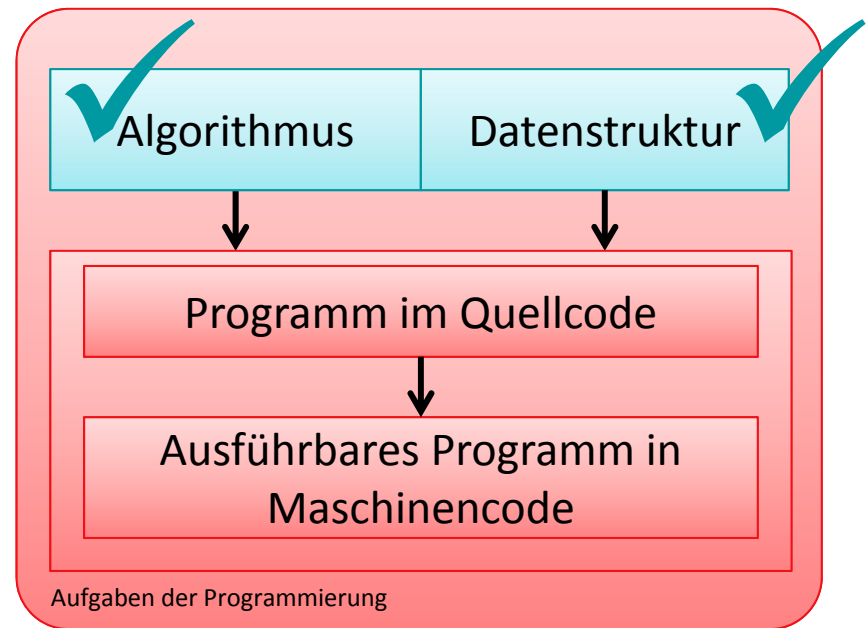
Wie wird der Programmcode der Programmiersprache in Maschinencode überführt?

- Compiler
- Interpreter

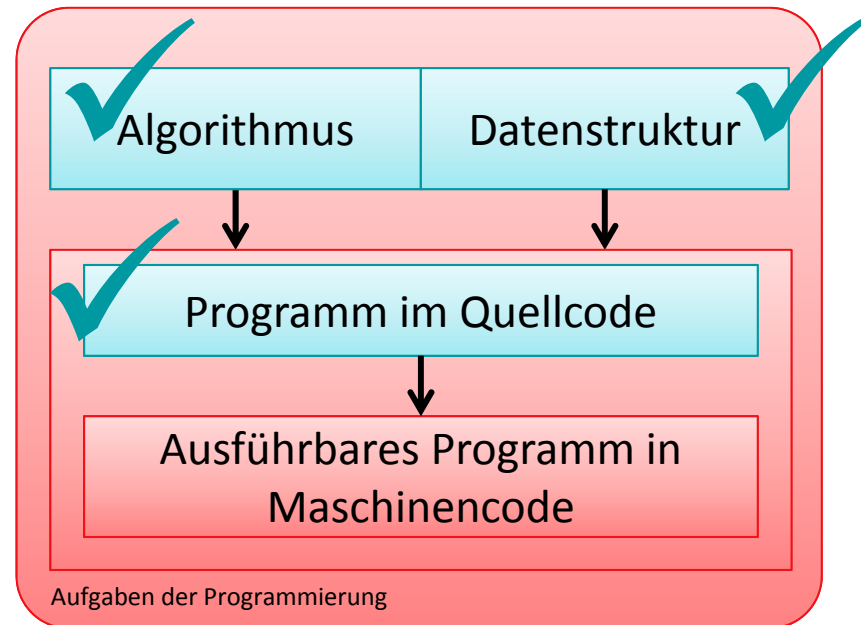
Algorithmus, Datenstruktur und Programm



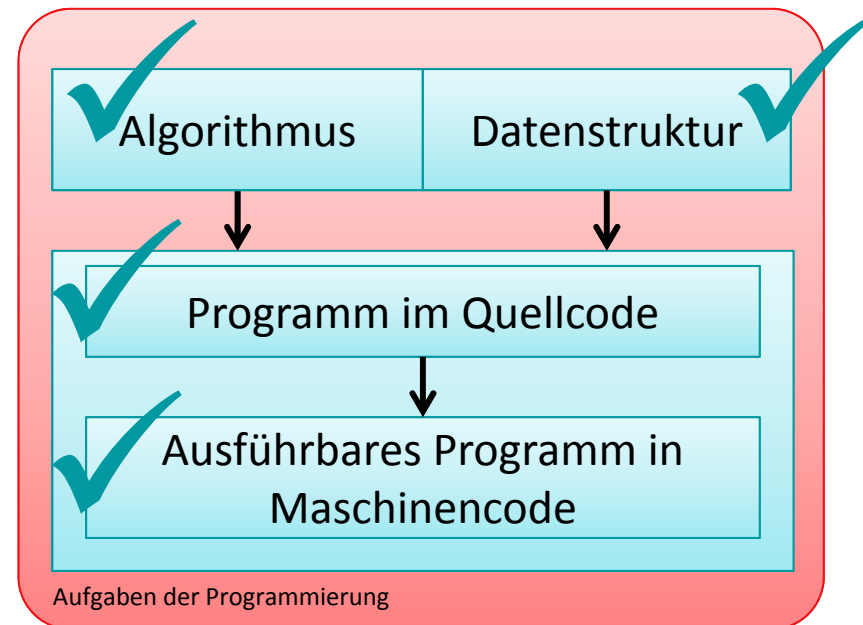
Algorithmus, Datenstruktur und Programm



Algorithmus, Datenstruktur und Programm



Algorithmus, Datenstruktur und Programm



Algorithmus, Datenstruktur und Programm



Im Vorfeld

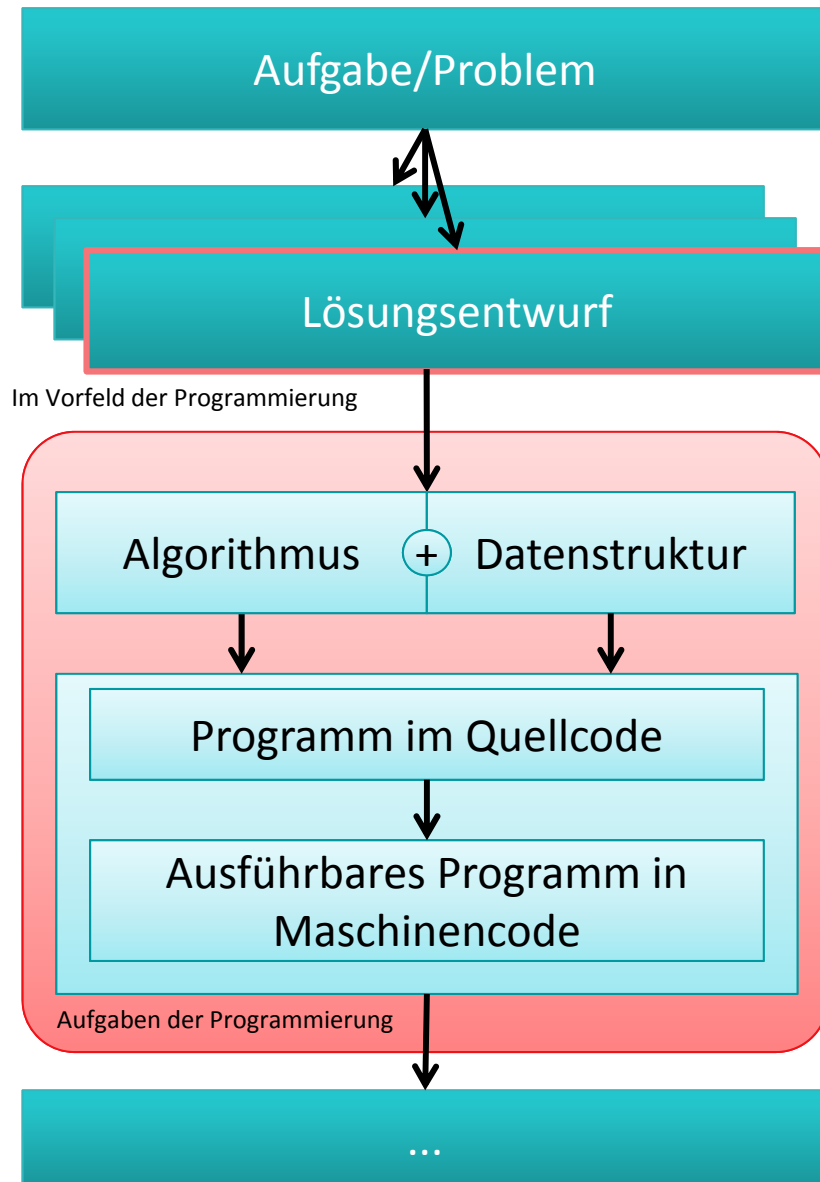
- Mindestens ein Lösungsentwurf für analysiertes Problem/Aufgabe

Stufenweise Umsetzung der Lösung

- Algorithmus
- Datenstruktur
- Quellcode
- Ausführbares Programm

Im Anschluss

- Test und weitere Aktivitäten





Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick

Arten von Programmiersprachen



Programmiersprachen

- werden fortlaufend verbessert, um neue Konzepte erweitert oder durch neue Sprachen ersetzt
- aktuell werden u.a. folgende Arten unterschieden
 - imperative Programmiersprachen
 - objektorientierte Programmiersprachen
 - deklarative Programmiersprachen
 - funktionale Programmiersprachen



Arten von Programmiersprachen

Imperative Programmiersprachen

- in die Programmiersprache überföhre Anweisungen des Algorithmus werden Schritt für Schritt abgearbeitet
- Anweisungen manipulieren Daten
- große oder komplexe Algorithmen werden in mehrere Unterprogramme (z.B. Prozeduren) strukturiert
- syn. prozedurale Programmiersprache
- Vertreter sind z.B. C, Pascal



Arten von Programmiersprachen

Objektorientierte Programmiersprachen

- in die Programmiersprache überföhre Anweisungen des Algorithmus werden durch die Interaktion von Objekten abgearbeitet
- Objekte
 - bilden Einheit von Daten und Anweisungen
 - sollen an die Dinge der realen Welt angelehnt sein
- Vertreter sind z.B. C++, Java

Arten von Programmiersprachen



Deklarative Programmiersprache

- beschreiben keinen Algorithmus, sondern das zu lösende Problem (was gelöst werden soll)
- der Computer legt selbst fest, in welcher Reihenfolge welche Anweisungen auszuführen sind, um das Problem zu lösen (wie es gelöst werden soll)
- Vertreter sind z.B. SQL, Prolog, QVT-R

Arten von Programmiersprachen



Funktionale Programmiersprache

- ausgehend vom mathematischen Konzept der Funktion, wird diese als Hauptelement der Programmiersprache verwendet
- Funktion bildet Eingabedaten auf Ausgabedaten ab
- anstelle des imperativen Sprachelements Schleife ruft in der funktionalen Programmierung eine Funktionen sich selbst wieder auf (Rekursion) bis Bedingung erfüllt ist
- Vertreter sind: Scala, XSLT, LISP

Arten von Programmiersprachen



Programmiersprachen

- werden fortlaufend verbessert, um neue Konzepte erweitert oder durch neue Sprachen ersetzt
- aktuell werden u.a. folgende Arten unterschieden
 - imperative Programmiersprachen
 - objektorientierte Programmiersprachen
 - deklarative Programmiersprachen
 - funktionale Programmiersprachen



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Entwicklungsumgebung

stellt eines oder mehrere Werkzeuge zur Programmentwicklung zur Verfügung

- Werkzeuge zur Erfassung von Quelltexten
- Integration von Compiler oder Interpreter
- Werkzeug für Fehlersuche (Debugger)
- Werkzeug zur Dokumentation
- Werkzeug zum Entwurf (auch zum grafischen Entwurf, d.h. Modellierung)
- Werkzeuge für die Gestaltung von Benutzeroberflächen (GUI-Design)
- ...

Beispiele: Visual Studio, Eclipse, JBuilder

Entwicklungsumgebung



NIST/ECMA-Referenzmodell (Toaster-Modell) von 1992

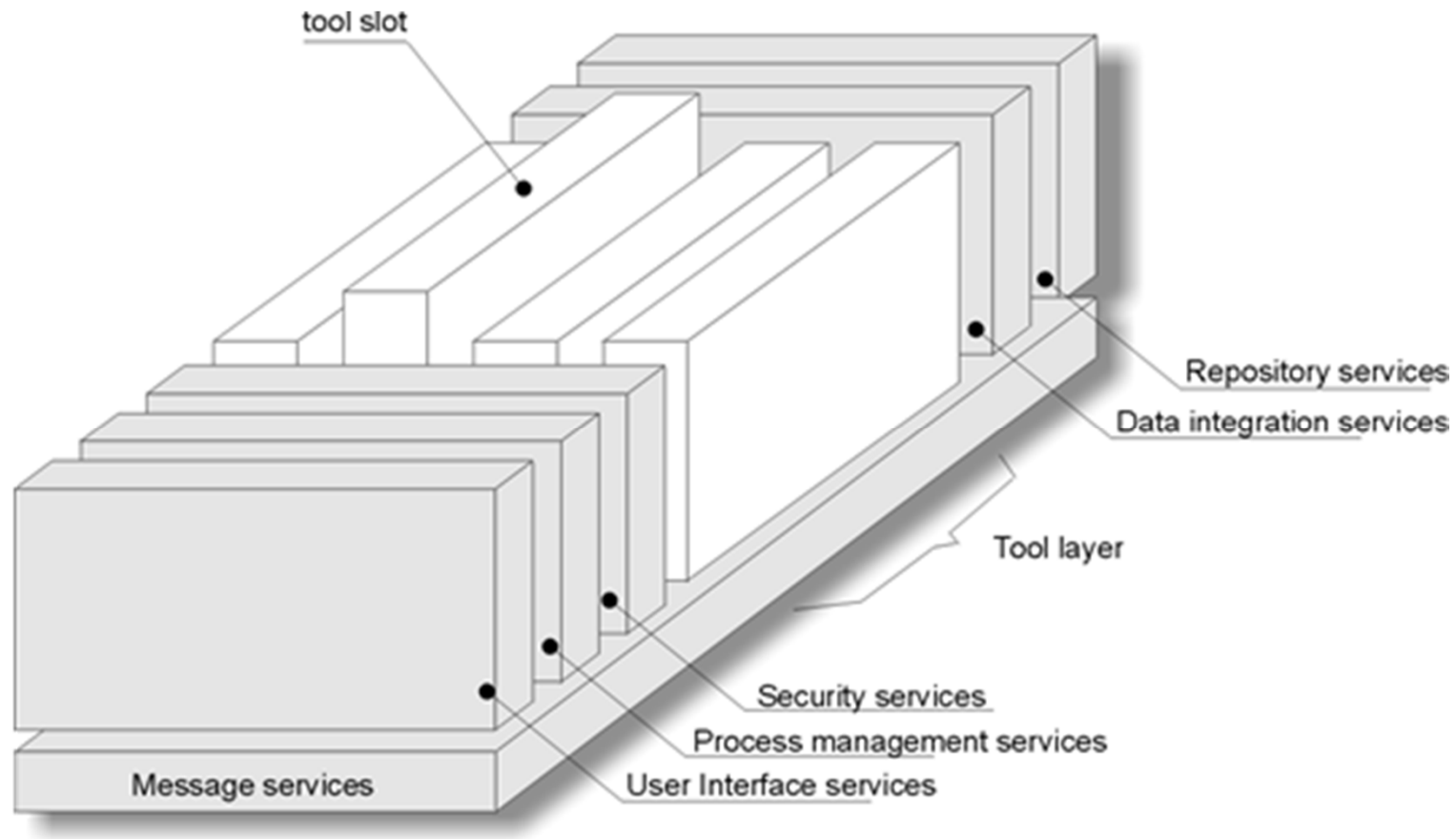


Abb. nach [Balzert, 1996], S. 593

Entwicklungsumgebung



Einordnung ausgewählter Bestandteile

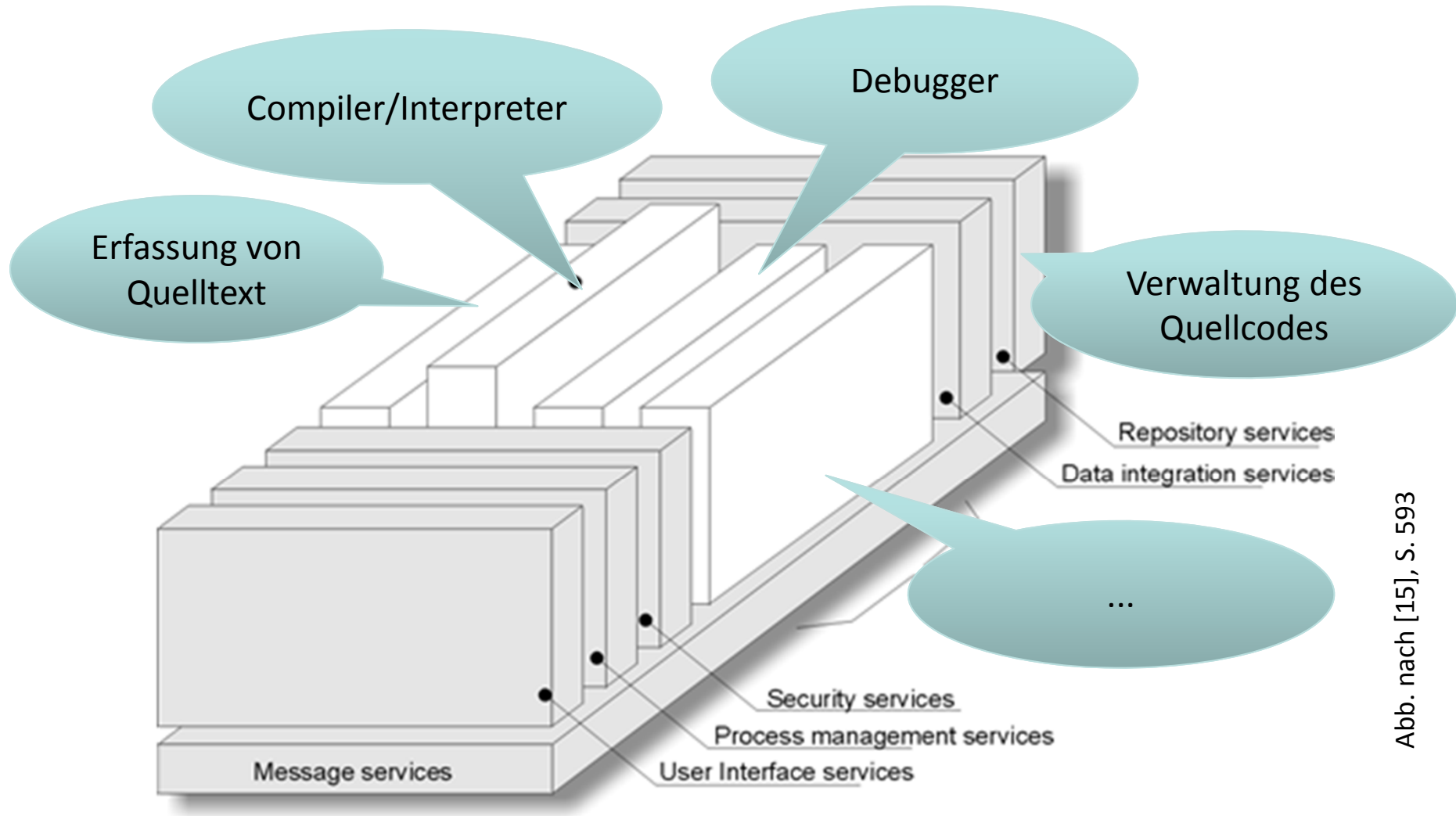


Abb. nach [15], S. 593

Entwicklungsumgebung



In diesem Kurs

- Verwendung von Microsoft Access mit der integrierten Programmiersprache Visual Basic for Applications (VBA)
- VBA-Editor ist die wesentliche Entwicklungsumgebung mit
 - Quellcodeerfassung und -verwaltung
 - Interpreter
 - Debugger
 - ...



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access/VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access/VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access/VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick

MS Access mit VBA als Programmierumgebung



Erste Schritte in MS Access mit VBA

- MS Access starten
- Leere Datenbank anlegen
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen

Erstes Programm

- Prozedur anlegen
- Einfache Ausgabe im Direktbereich
- Prozedur ausführen

MS Access mit VBA als Programmierumgebung



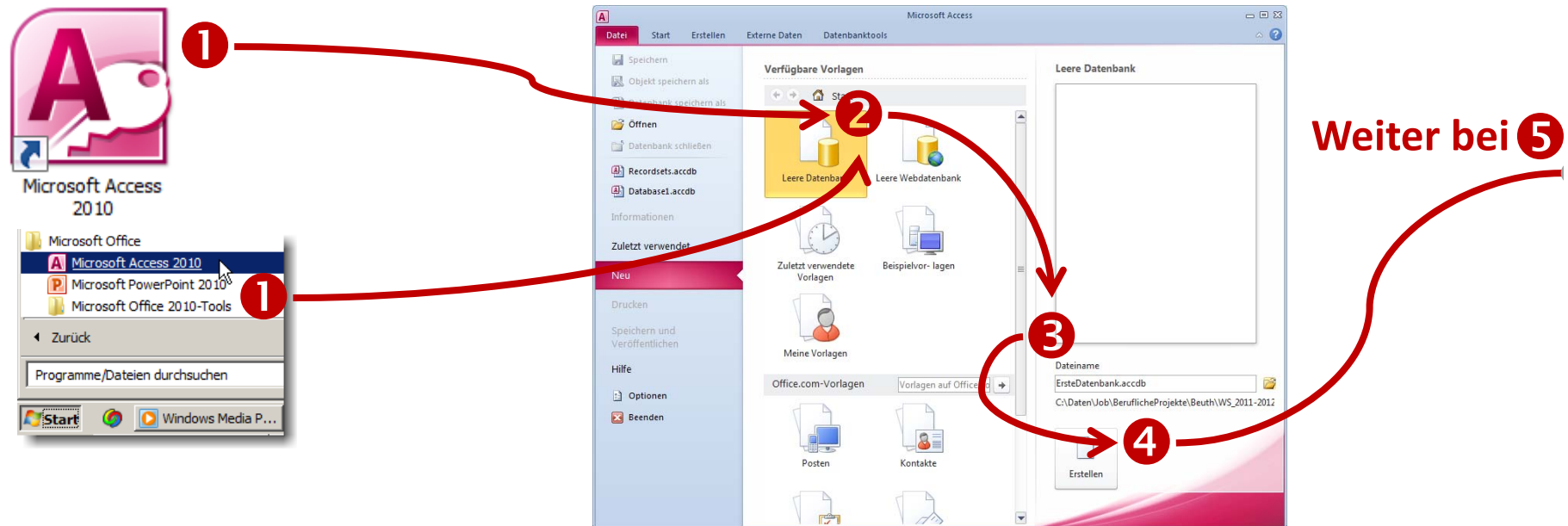
Erste Schritte in MS Access mit VBA

- MS Access starten
- Leere Datenbank anlegen
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen

Erstes Programm

- Prozedur anlegen
- Einfache Ausgabe im Direktbereich
- Prozedur ausführen

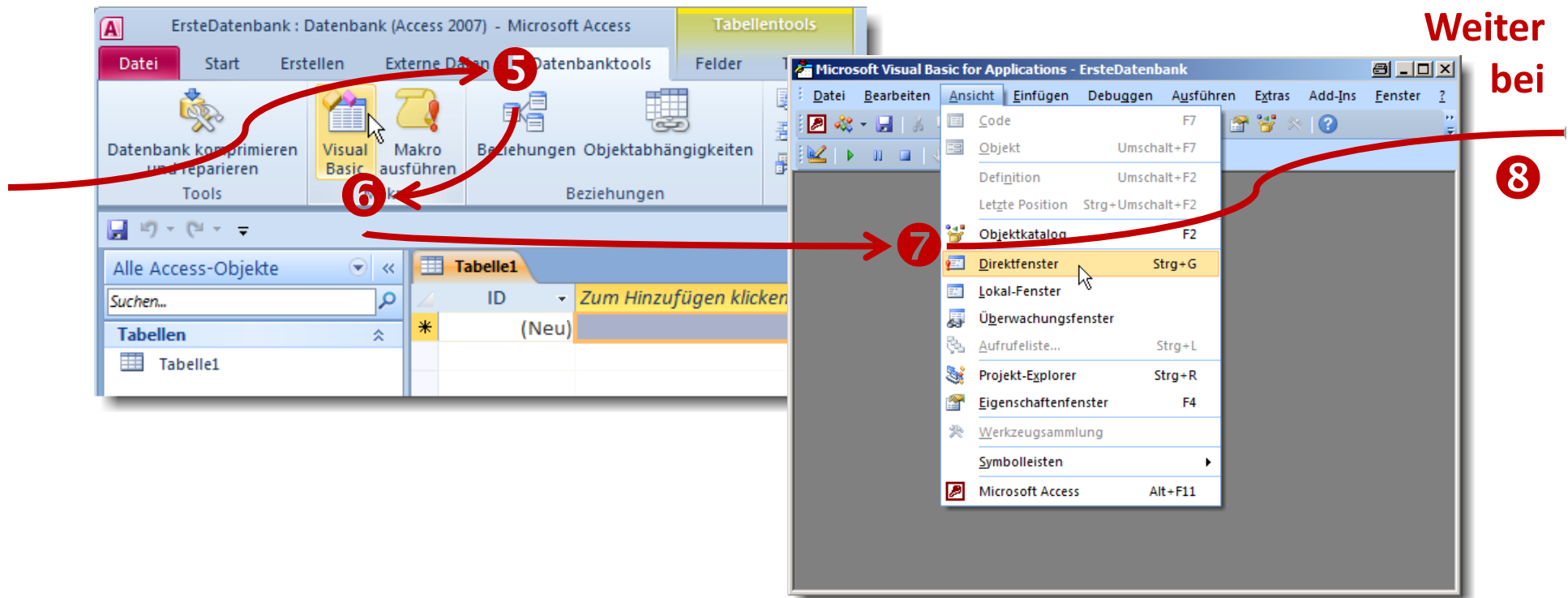
Beispiel - Erste Schritte in MS Access mit VBA (1)



- **Starten von MS Access**
 - z.B. Desktop-Icon oder
 - z.B. über Menü Start>Alle Programme>Microsoft Office

- **Leere Datenbank anlegen**
 - bei verfügbaren Vorlagen "Leere Datenbank wählen"
 - im rechten Fensterbereich im Feld "Dateiname" einen Dateinamen erfassen
 - Schaltfläche "Erstellen" betätigen

Beispiel - Erste Schritte in MS Access mit VBA (2)

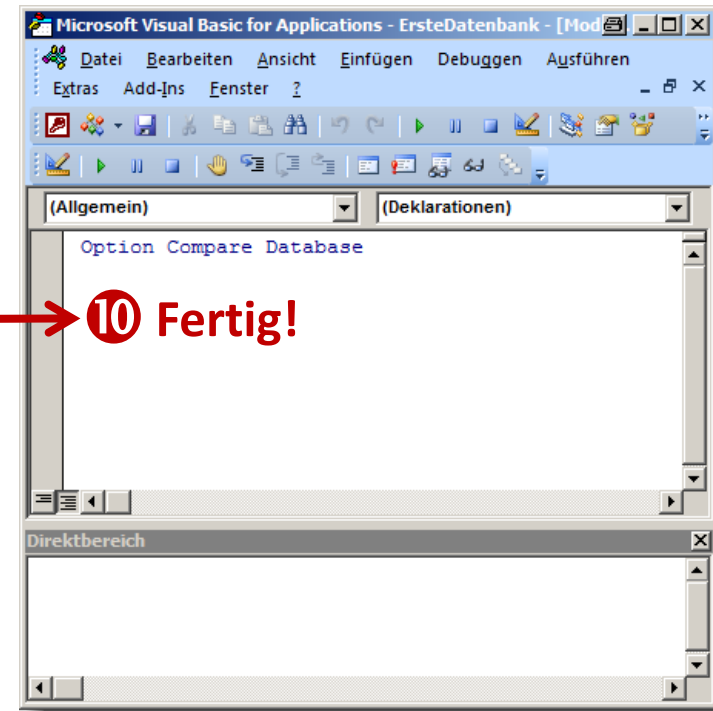
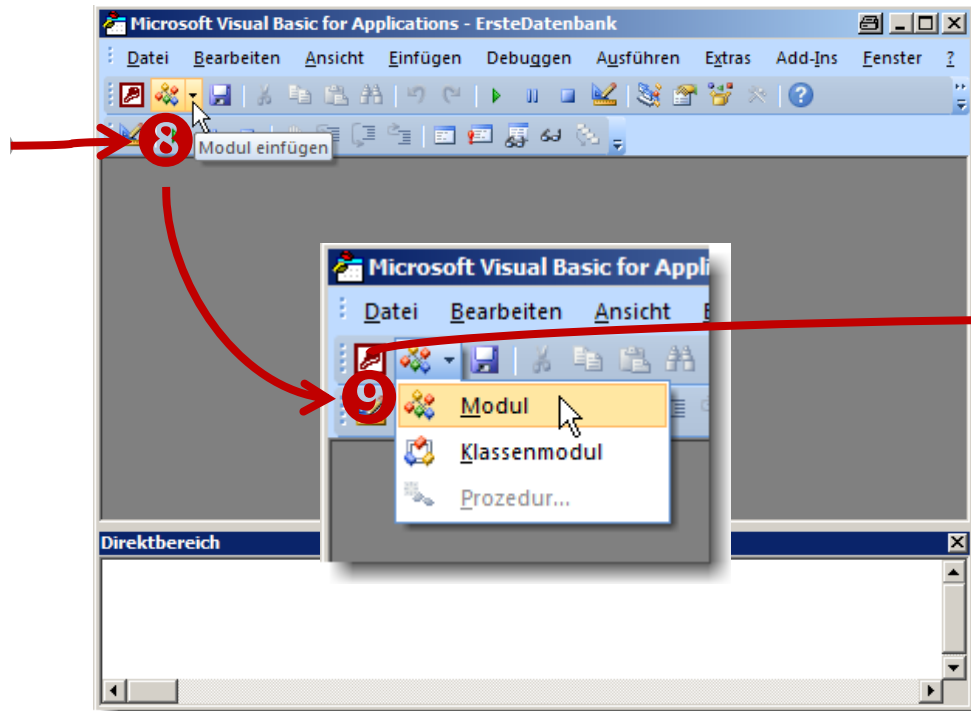


- **In neu angelegter, leerer Datenbank**
 - Registerkarte "Datenbanktools" auswählen
 - Icon "Visual Basic" wählen

Im geöffneten "Visual Basic for Applications"

- Menü Ansicht>Direktfenster einblenden (Strg+G)

Beispiel - Erste Schritte in MS Access mit VBA (3)



- Symbolleistenicon "Modul einfügen" aufklappen (per "▼")
- Option "Modul" auswählen

Neues Modul angelegt,
Eingabemarkierung blinkt im
Editorbereich

Beispiel - Erste Schritte in MS Access mit VBA (4)



Prinzipieller Ablauf

- MS Access gestartet (per Desktop-Icon oder Start-Menü)
- Leere Datenbank in Access angelegt
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen

MS Access mit VBA als Programmierumgebung



Erste Schritte in MS Access mit VBA

- MS Access starten
- Leere Datenbank anlegen
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen

Erstes Programm

- Prozedur anlegen
- Einfache Ausgabe im Direktbereich
- Prozedur ausführen

MS Access mit VBA als Programmierumgebung



Erste Schritte in MS Access mit VBA

- MS Access starten
- Leere Datenbank anlegen
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen

Erstes Programm

- Prozedur anlegen
- Einfache Ausgabe im Direktbereich
- Prozedur ausführen

Beispiel - Erstes Programm in VBA (1)



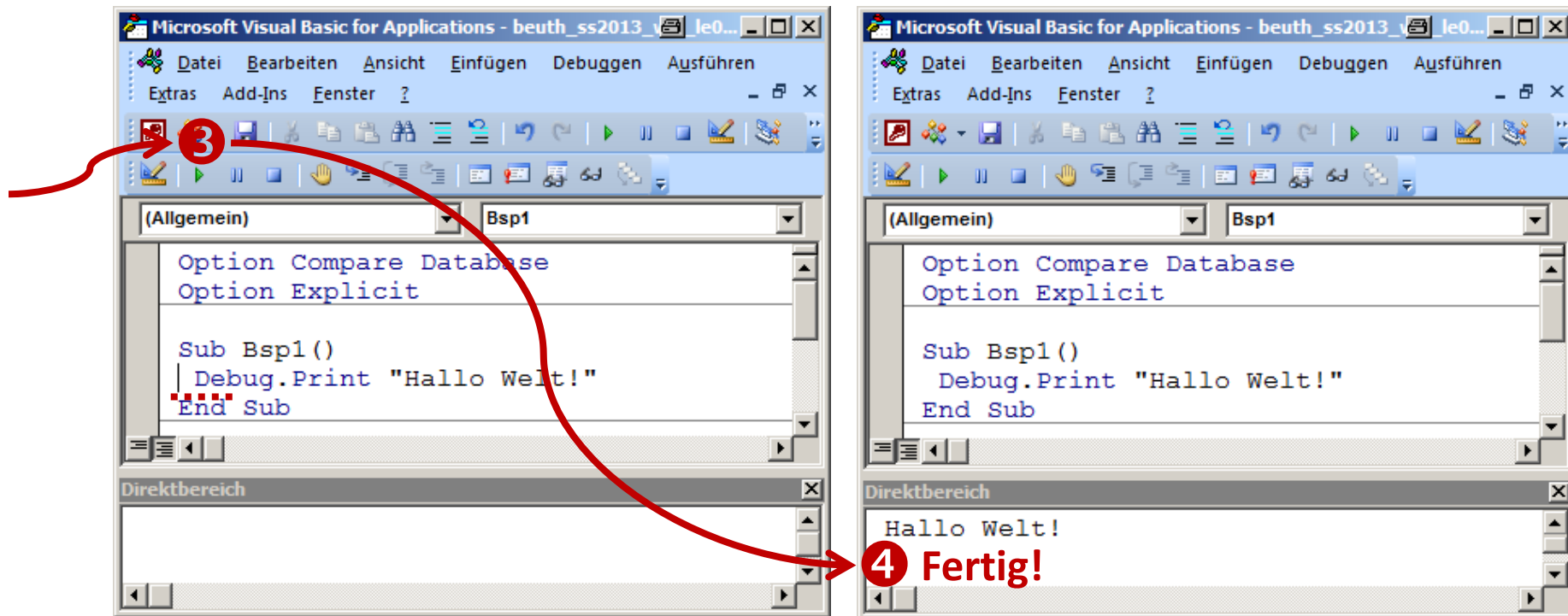
```
Sub Bsp1()  
    Debug.Print "Hallo Welt!"  
End Sub
```

Ausgangspunkt

- geöffnetes Fenster "Microsoft Visual Basic for Application"
- Eingabemarkierung im Editor

Erfassen der ersten Prozedur "Bsp1"

Beispiel - Erstes Programm in VBA (2)



Ausführen der Prozedur

- Eingabemarkierung muss innerhalb der Prozedur (d.h. zwischen Sub und End Sub) stehen
- Ausführen per Icon "▶" oder
- Funktionstaste F5

Ausgabe erfolgt im Direktbereich

Beispiel - Erstes Programm in VBA (3)



Neue Prozedur erfasst

- Schlüsselwort Sub leitet den Beginn einer Prozedur ein
- es folgt ein Namen der Prozedur (Bezeichner), der innerhalb des Moduls eindeutig sein muss
- Schlüsselwort End Sub schließt eine Prozedur ab

Anweisung Debug.Print dient zur Ausgabe im Direktbereich

An Anweisung schließt sich der auszugebende Textes an

Ausführen der Prozedur

- Eingabemarkierung muss innerhalb der Prozedur positioniert sein
- Per Menüeintrag Ausführen>Sub/User Form ausführen, per "Play"-Icon ► oder Funktionstaste F5

MS Access mit VBA als Programmierumgebung



Erste Schritte in MS Access mit VBA

- MS Access starten
- Leere Datenbank anlegen
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen

Erstes Programm

- Prozedur anlegen
- Einfache Ausgabe im Direktbereich
- Prozedur ausführen

MS Access mit VBA als Programmierumgebung



Erste Schritte in MS Access mit VBA

- MS Access starten
- Leere Datenbank anlegen
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen

Erstes Programm

- Prozedur anlegen
- Einfache Ausgabe im Direktbereich
- Prozedur ausführen



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick

Zusammenfassung



Grundkonzepte/-begriffe

- Algorithmus
 - Definition: präzise, vollständige, eindeutig formulierte, endliche Verarbeitungsvorschrift, die Ausgangssituation in ein Ergebnis überführt, das zur Lösung einer Aufgabe dienen soll.
 - Bestandteile: Anweisungen, Ablauf, Verzweigungen, Schleifen, Unterprogramme
 - Beschreibung: Struktogramme, Programmablaufpläne, UML-Aktivitätsdiagramme, ...
- Datenelemente und -strukturen
 - einfache Datenelemente, zum Schreiben und Lesen eines Werts;
 - komplexe Datenelemente, die aus einfachen aufgebaut sind und
 - komplexe Datenstrukturen, die Datenelemente in bestimmter Form organisieren und außer schreibendem und lesendem Zugriff spezielle Aktionsmöglichkeiten bieten (z.B. Einfügen, Entfernen).
 - Beispiele: Verkettete Liste, Stapel, Schlange, Baum

Zusammenfassung



Grundkonzepte/-begriffe

- Programm
 - mit den Sprachmitteln einer konkreten Programmiersprache ausgedrückter Algorithmus in Verbindung mit den ebenso ausgedrückten Datenstrukturen zur Ausführung in einem Computer
 - kann vorliegen als
 - Quellcode: Darstellung in einer lesbaren und verständlichen Programmiersprache
 - Maschinencode: Darstellung mit Befehlen aus dem Befehlsvorrats des konkret verwendeten Computers
- Programmiersprache: Formale Sprache zur Formulierung von Programmen mit präziser Syntax und eindeutiger Semantik
- Maschinensprache: Binäre und ausführbare Darstellung des Programms, abhängig von der verwendeten Hardware

Zusammenfassung



Grundkonzepte/-begriffe

- Compiler
 - fertiger Programmcode wird vollständig dem Compiler übergeben und
 - über ein Zwischenformat in ausführbares Programm übersetzt, das geladen und ausgeführt werden kann
- Interpreter
 - liest eine Anweisung aus dem Quellcode und übersetzt sie in Maschinencode, lädt die Anweisung und führt sie aus
 - dann wird mit der nächsten Anweisung fortgefahren



Zusammenfassung

Programmiersprachen

- werden fortlaufend verbessert, um neue Konzepte erweitert oder durch neue Sprachen ersetzt
- aktuell werden beispielsweise imperative, objektorientierte, deklarative und funktionale Programmiersprachen

Entwicklungsumgebung

- stellt eines oder mehrere Werkzeuge zur Programmentwicklung zur Verfügung (z.B. zur Erfassung von Quellcode, den Compiler oder Interpreter, Debugger, Werkzeuge für die Gestaltung von Benutzeroberflächen)
- hier Verwendung von Microsoft Access mit der integrierten Programmiersprache Visual Basic for Applications (VBA)

Zusammenfassung



Prinzipieller Ablauf in MS Access/VBA

- MS Access gestartet (per Desktop-Icon oder Start-Menü)
- Leere Datenbank in Access angelegt
- Visual Basic Editor for Applications öffnen
- Neues Modul anlegen
- Neue Prozedur anlegen (Sub + Bezeichner + End Sub)
- Anweisung innerhalb der Prozedur erfassen
 - Beispiel: Debug.Print für die Ausgabe im Direktfenster
- Ausführen der Prozedur
 - Eingabemarkierung muss innerhalb der Prozedur positioniert sein
 - Per Menüeintrag Ausführen>Sub/User Form ausführen, per "Play"-Icon ▶ oder Funktionstaste F5

Zusammenfassung



Wir haben bereits folgende Elemente eines Programms kennengelernt

- Anweisungen: Debug.Print
- Schlüsselwörter (reservierte Wörter)
 - Sub
 - End Sub
- Bezeichner
 - Bsp1
- Literale, z.B. "Hallo Welt"
- ...



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick



Inhalt

Rückblick

Einordnung und Grundkonzepte

- Algorithmus
- Datenstruktur
- Programm

Arten von Programmiersprachen

- Imperative Sprachen
- Objektorientierte Sprachen
- Deklarative Sprachen
- Funktionale Sprachen

Entwicklungsumgebung

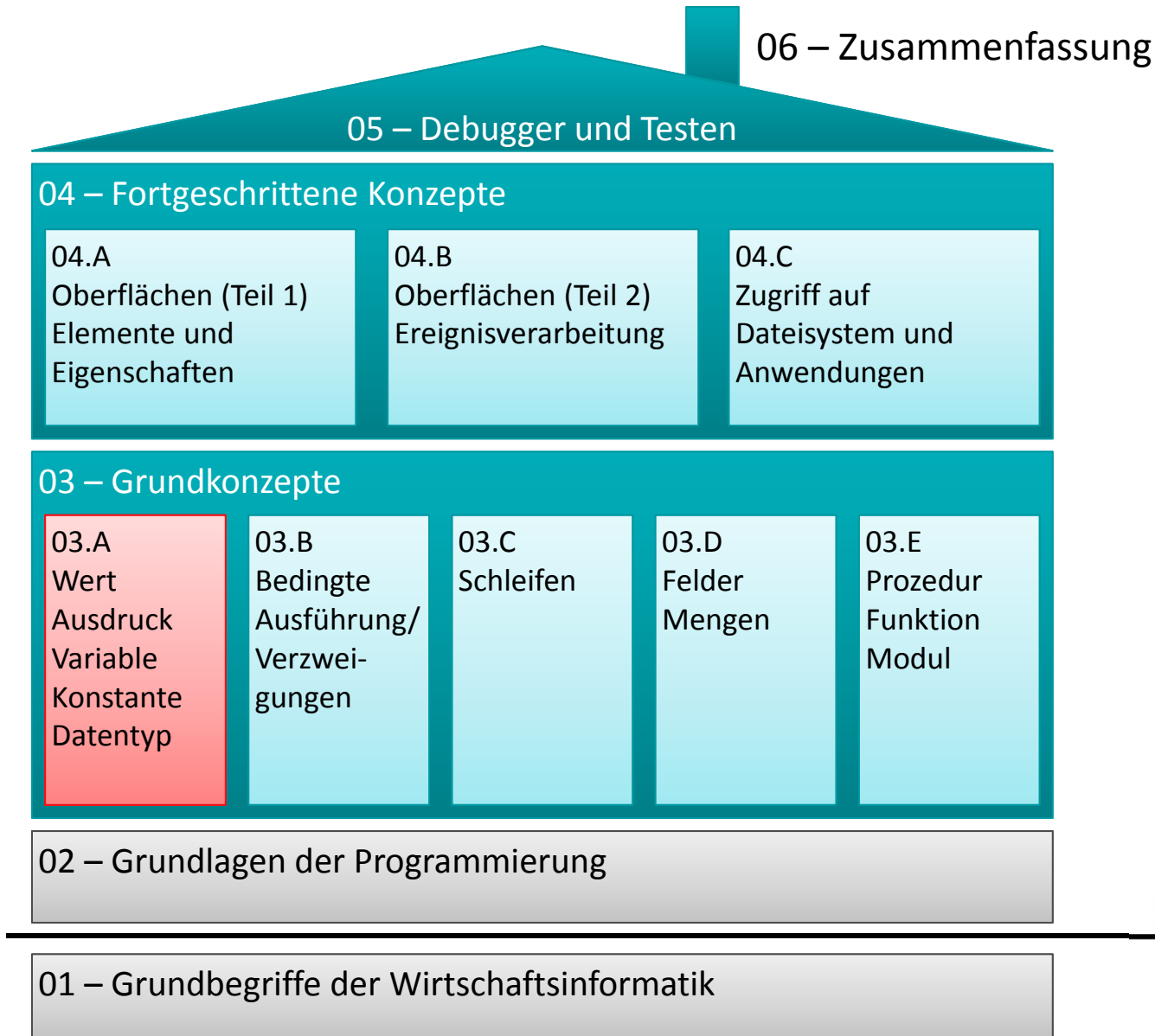
MS Access und VBA als Programmierumgebung und -sprache

- Anweisungen
- Ausgabe

Zusammenfassung

Ausblick

Ausblick



Literatur



[Balzert, 1996] H. Balzert: Lehrbuch der Softwaretechnik, Spektrum Akad. Verlag (1996)

[Broy, 1992] M. Broy: Informatik. Eine Grundlegende Einführung. Teil 1 Problemnahe Programmierung; Berlin, u.a; Springer; 1992

[Fink et al., 2001] A. Fink, G. Schneiderei, S. Voß: Grundlagen der Wirtschaftsinformatik. Physica-Verlag, Heidelberg (2001).

[Krallmann , 1996] H. Krallmann: Systemanalyse im Unternehmen. Oldenbourg, 2. Aufl. (1996).

[Lehner et al., 2008] Franz Lehner, Stephan Wildner, Michael Scholz: Wirtschaftsinformatik : Eine Einführung. Hanser, München, 2. Aufl. (2008)[Duden, 2001] Duden Informatik A-Z. Fachlexikon für Studium, Ausbildung und Beruf. Bibliographisches Institut, Mannheim, 3. Aufl. (2001)



Quellen

- [1] Quelle: Frank C. Müller,
http://de.wikipedia.org/wiki/Datei:Kaffeemaschine_fcm.jpg, Lizenz: Creative Commons-Lizenz Namensnennung-Weitergabe unter gleichen Bedingungen 2.0 Deutschland
- [2] Yomi955 aus Kyoto,
http://de.wikipedia.org/wiki/Datei:Sunny_side_up_by_yomi955.jpg, Lizenz: Creative Commons-Lizenz Namensnennung 2.0 US-amerikanisch
- [3] Wikipedia: Begriff Pseudocode, <http://de.wikipedia.org/wiki/Pseudocode>, Aufruf am 12.03.2013
- [4] Wikipedia: Begriff Struktogramm, <http://de.wikipedia.org/wiki/Struktogramm>, Aufruf am 12.03.2013
- [5] Robert Kosara,
http://de.wikipedia.org/wiki/Datei:Ben_Shneiderman_at_UNCC.jpg, Lizenz: Creative Commons CC0 1.0 Verzicht auf das Copyright
- [6] Isaac Nassi: Private Homepage. http://www.nassi.com/Nassi-1_hires_8x10_15Aug2007.jpg , Aufruf am 12.03.2013
- [7] Online-Skript zur Lehrveranstaltung "Grundlagen der Programmierung". WS 2011/12, FB I, Beuth Hochschule für Technik Berlin



BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Wirtschaftsinformatik 1

LE 02 – Grundlagen der Programmierung

Prof. Dr. Thomas Off

<http://www.ThomasOff.de/lehre/beuth/wi1>