Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Grundlagen der Programmierung & Algorithmen und Datenstrukturen

Einführung

Die Inhalte der Vorlesung wurden primär auf Basis der angegebenen Literatur erstellt. Darüber hinaus orientiert sich diese an der Vorlesung von Prof. Dr.-Ing. Faustmann (ebenfalls HWR Berlin).

Überblick zur Softwaretechnik

Begriff der Softwaretechnik

In der Softwaretechnik wird Software nach bestimmten Vorgehensweisen erstellt. Diese Vorgehensweisen werden in einem konkreten Softwareprozess (software life cycle) umgesetzt.

Zeit, Kosten, Qualität

Vorgehensmodelle

Definition SW-Technik:

Zielorientierte Bereitstellung und systematische Verwendung von Prinzipien, Methoden, Konzepten, Notationen und Werkzeugen für die arbeitsteilige, ingenieurmäßige Entwicklung und Anwendung von umfangreichen SW-Systemen.

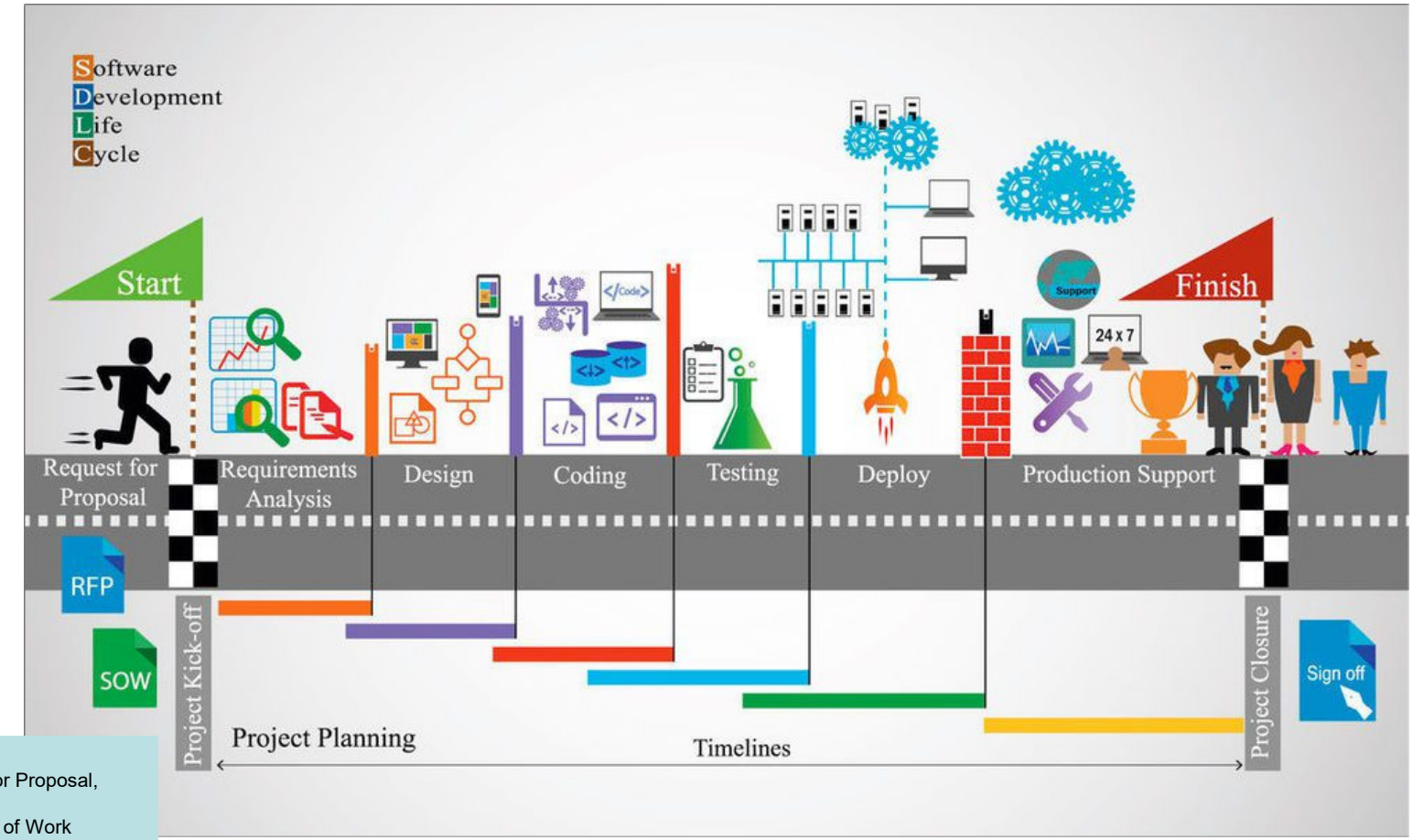
z.B. Objektorientierung

z.B. UML, Java

z.B. Konfigurations-management

Allgemein findet man in jedem Vorgehensmodell zur Softwareerstellung die Phasen Analyse, Entwurf, Codierung, Test und Betrieb/Wartung.

Software-Lebenszyklus



RFP
Request for Proposal,
SOW
Statement of Work

Quelle der Abb.: Matzer, M.; Augsten, S.: Software-Lebenszyklus nach Standard, (Asha Sreenivas - stock.adobe.com), https://www.dev-insider.de/software-lebenszyklus-nach-standard-122072017-a-697220/, Abruf: Oktober 2020

„Lastenheft“ vs. „Pflichtenheft“

- Lastenheft (DIN 69905) - „vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines Auftrages“.
- Pflichtenheft (DIN 69905) - „vom Auftragnehmer erarbeiteten Realisierungsvorgaben aufgrund der Umsetzung des vom Auftraggeber vorgegebenen Lastenhefts“

Übung Softwaretechnik

- Nennen Sie Aspekte/Aufgabenstellungen im Entwurf, die in der Analysephase keine Rolle spielen.
- Können Sie auch für den umgekehrten Fall entsprechende Aspekte nennen? Begründen Sie Ihre Antwort!
- Es soll eine Software für ein Mietwagenunternehmen erstellt werden. Es werden verschiedene Klassen von Autos zu den gewünschten Zeiten (Stunden, Tage, Wochen) vermietet. Das Unternehmen stellt das Fahrzeug zur gewünschten Zeit an einem definierten Übergabeort zur Verfügung. Der Inhaber möchte für die Planung, Verwaltung und Rechnungslegung seiner Aufträge Unterstützung erhalten.
  - → Beschreiben Sie mögliche Ergebnisse der verschiedenen Phasen eines solchen Softwareprojekts.
  - → Gruppenarbeit 30 min. (je 5 Teilnehmer)
  - → Vorstellung von 2 Gruppenergebnissen in max. 5 min.

Spezifikation und Algorithmen

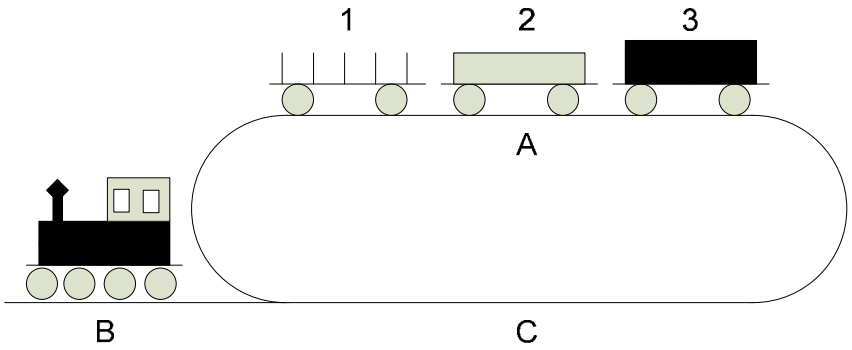
Spezifikation

Eine Spezifikation ist eine vollständige, detaillierte und unzweideutige Problembeschreibung. Dabei heißt eine Spezifikation

- *Vollständig*, wenn alle Anforderungen und alle relevanten Rahmenbedingungen angegeben worden sind
- *Detailliert*, wenn klar ist, welche Hilfsmittel, insbesondere welche Basis-Operationen zur Lösung zugelassen sind
- *Unzweideutig*, wenn klare Kriterien angegeben werden, wann eine Lösung akzeptabel ist

Quelle: Gumm, H. P.; Sommer, M.: Einführung in die Informatik, 8. Auflage. München, Oldenbourg Verlag, 2009

Spezifikation



Informelle Spezifikation – Rangierproblem:

„Eine Lokomotive soll die im Gleisabschnitt A befindlichen Wagen 1, 2, 3 in der Reihenfolge 3, 1, 2 auf Gleisstück C abstellen“

- Bewerten Sie die Vollständigkeit, Detailliertheit und Unzweideutigkeit die aufgezeigten Problembeschreibung!

→ Einzelüberlegung 5 min.

→ Gemeinsame Diskussion

Quelle: Gumm, H. P.; Sommer, M.: Einführung in die Informatik, 8. Auflage. München, Oldenbourg Verlag, 2009

Spezifikation

Berechnung des größten gemeinsamen Teiler zweier Zahlen:

Potentielle informelle Spezifikation:

> *Für beliebige Zahlen M und N berechne den größten gemeinsamen Teiler ggT(M,N), also die größte Zahl, die sowohl M als auch N teilt.*

Ein informelle Spezifikation lässt viele Fragen offen:

> *Vollständigkeit: Welche Zahlen M, N sind zugelassen? Dürfen M und N nur positive Zahlen oder auch negative oder gar rationale Zahlen sein? Ist 0 erlaubt?*
>
> *Detailliertheit: Welche Operationen sind erlaubt? (+, -, div, mod)*
>
> *Unzweideutigkeit: Was heißt berechnen? Soll das Ergebnis ausgedruckt werden oder in einer bestimmten Variablen gespeichert werden?*

Quelle: Gumm, H. P.; Sommer, M.: Einführung in die Informatik, 5. Auflage. München, Oldenbourg Verlag, 2002

Spezifikation

Verwendung von Vor- und Nachbedingungen durch Angabe eines Paares P und Q von logischen Aussagen. {P} A {Q}.

„Wenn der Algorithmus A in einer Situation gestartet wird, in der P gilt, dann wird, wenn A beendet ist, Q gelten.“

Beispiel ggt(a,b):

Vorbedingung

`{a und b sind ganze Zahlen mit 0<a<32767 und 0<b<32767}`

Nachbedingung

`{z = ggt(a,b), d.h., z ist Teiler von a und b und für jede andere Zahl z‘, die auch a und b teilt, gilt z‘≤z}`

Quelle: Gumm, H. P.; Sommer, M.: Einführung in die Informatik, 8. Auflage. München, Oldenbourg Verlag, 2009

Algorithmen

Detaillierte und explizite Vorschrift zur schrittweisen Lösung eines Problems.
D.h. im einzelnen:

- Die Ausführung des Algorithmus erfolgt in einzelnen Schritten.
- Jeder Schritt besteht aus einer einfachen und offensichtlichen Grundaktion.
- Zu jedem Zeitpunkt muss eindeutig definiert sein, welche Schritte als nächste auszuführen sind.

„Ein Algorithmus kann daher sowohl von einem Menschen oder von einer Maschine durchgeführt werden. Ist der jeweils nächste Schritt bekannt spricht man von einem deterministischen, ansonsten von einem nichtdeterministischen Algorithmus.“

Quelle: Gumm, H. P.; Sommer, M.: Einführung in die Informatik, 8. Auflage. München, Oldenbourg Verlag, 2009

Algorithmen

Sichten auf Algorithmen

Ein Algorithmus löst typischerweise eine ganze Klasse von Problemen. Das jeweils zu bearbeitende Problem wird durch Parameter bestimmt. Ein Algorithmus liefert zu einer solchen Eingabe jeweils ein Resultat.

Repräsentationsformen:

- informell
- formal

Funktionale Sicht: Welche Aufgabenklasse soll der Algorithmus bewältigen – vgl. mathem. Funktionsbegriff: y = f(x)?

Operationale Sicht: Wie sollen die Aufgaben bewältigt werden?

- Angabe von elementaren Verarbeitungsschritten
- Beschreibung des Kontrollflusses
- Angabe von Daten und Parametern

Algorithmen

Der Euklidsche Algorithmus (3. Jh. V. Chr.) zur Berechnung des größten gemeinsamen Teilers zweier natürlicher Zahlen.

Eine informelle Beschreibung

Aufgabenstellung:

Gegeben: Zwei ganze Zahlen a und b mit a>0 und b>0.

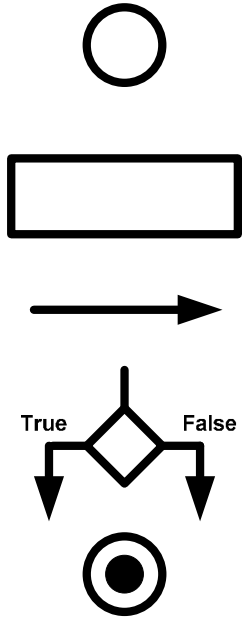
Gesucht: Der größte gemeinsame Teiler *ggt(a,b)* von a und b.

Der Algorithmus *ggt(a,b):*

- (1) Falls *a gleich b* ist, gilt ggt(a,b) = a;
- (2) Falls *a kleiner als b*, wende *ggt* auf *a* und *b-a* an;
- (3) Falls *a größer b* ist, wende *ggt* auf *a-b* und *b* an.

Übung Algorithmen

- Entwickeln Sie mit Hilfe der grafischen Notation für Fluss-diagramme einen Algorithmus zur Lösung des ggT-Problems.
- Einsatz der folgenden Notationselemente:
  - Beginn der Ausführung - Kreis
  - Elementaraktion – Rechteck
  - Sequenz des Algorithmus - Pfeile
  - Bedingte Verzweigung – True/False
  - Ende des Algorithmus – gefüllter Kreis



→ Bearbeitung 20 min.

Algorithmen

Für den *ggt(a,b)* Algorithmus gilt:

- Die arithmetische Operation „-“ und die Vergleichsoperationen „kleiner als“, „größer als“ und „gleich“ werden als elementare Verarbeitungsschritte vorausgesetzt.
- Der Algorithmus ist **deterministisch**, d.h. für jede Eingabe ist die Folge der einzelnen Schritte genau festgelegt.

  Für deterministische Algorithmen ist die Relation zwischen Eingabe- und Ausgabewerten eine partielle Abbildung.

  Der Algorithmus heißt **korrekt**, wenn diese Funktion der Aufgabenstellung entspricht, die der Algorithmus bewältigen soll.

Algorithmen

Algorithmen werden u.a. durch folgende Merkmale klassifiziert. Ein Algorithmus heißt

- **terminierend**, wenn er für alle zulässigen Schrittfolgen nach endlich vielen Schritten endet,
- **deterministisch**, wenn die Abfolge der einzelnen Verarbeitungsschritte eindeutig festgelegt ist,
- **determiniert**, wenn es zu jeder Eingabe höchstens eine Ausgabe gibt, d.h. der Algorithmus berechnet eine (partielle) Funktion,
- **endlich,** wenn er mit einer endlichen Anzahl von Worten beschrieben werden kann,
- **sequentiell**, wenn die Verarbeitungsschritte stets hintereinander ausgeführt werden,
- **parallel**, wenn gewisse Verarbeitungsschritte nebenläufig ausgeführt werden.

Übung Algorithmen

- Wann kann man einen Algorithmus als korrekt bezeichnen? Wie könnte man diese Eigenschaft nachweisen?
- In welchem Bezug stehen Algorithmen und Programme zueinander? Entscheiden Sie für die folgenden Bereiche, ob es sich um einen Algorithmus oder/und ein Programm (oder um keines von beiden) handelt. Begründen Sie ihr Ergebnis:
  - Ergebnis einer Routenplaneranfrage im WWW
  - Wahlprogramm einer Partei
  - Backanleitung für den Brotbackautomat
  - Verhalten eines kundenfreundlichen Fahrscheinautomaten
  - Musiknoten auf einem Notenblatt
- Warum ist der als Beispiel gegebene Algorithmus (ggt) deterministisch?

→ Gruppenarbeit 20 min. (je 4 Teilnehmer)

→ Vorstellung von 2 Gruppenergebnissen in max. 5 min.

Programmiersprachen und Programmierparadigmen

Der Programmbegriff

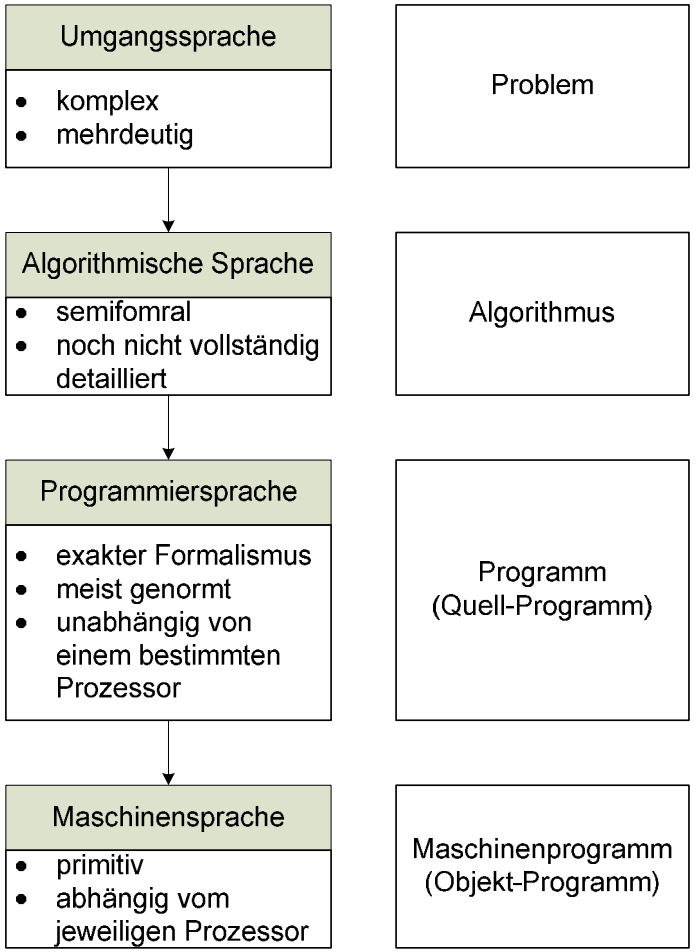
Algorithmen, die von einem automatischen Prozessor abgearbeitet werden, bezeichnet man als Programme. Ein Programm stellt die Realisierung eines Algorithmus dar. Im Gegensatz zu einem Algorithmus ist ein Programm konkreter und eingeschränkter.

- Es wird eine Programmiersprache gewählt, die auf der Basis definierter Grundelemente den Algorithmus vollständig beschreibt.
- Für benötigte Daten wird eine bestimmte Darstellung (Repräsentation – allgemeiner Begriff der Datentypen) gewählt.

Quelle: Balzert, H.: Lehrbuch Grundlagen der Informatik, Spektrum Adademischer Verlag, Heidelberg/Berlin 1999

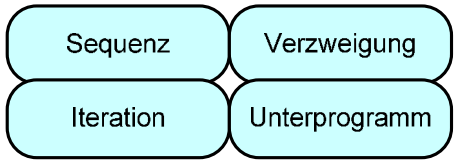
Programmieren und Programmiersprachen

- Kommunikation „Mensch – Maschine“
- Unterschiede im Sprachvorrat
- Mensch - Umfangreicher Sprachvorrat
- Maschine - Eingeschränkter Sprachvorrat
  1. Binäre Maschinensprachen
  2. Assemblersprachen – Mnemonik der Befehle
  3. Hochsprachen: (imperative, objektorientiert, …)
  4. Anwendungsorientierte 4GLs (Macro- und Scipt-Sprachen)
  5. Deklarative und logische Ansätze (z.B. SQL oder Prolog) oder auch NLP-Ansätze

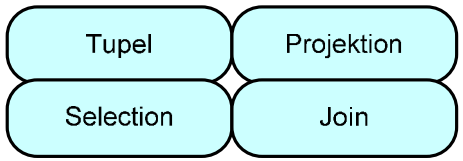


Programmierparadigmen

- Imperatives Programmierparadigma (z.B. BASIC, Pascal, Modula, Java, C/C++, C#, SMALLTALK) → Fokus auf das WIE
  - Strukturierte Programmierung
  - Prozedurale Programmierung
  - Objektorientierte Programmierung



- Deklarative Programmierparadigma (z.B. Haskell, LISP, Python, SQL, XML/HTML) → Fokus auf das WAS
  - Logische Programmierung
  - Constraint-Programmierung
  - Funktionale Programmierung



Fokus: SQL Abfragesprache

- Vergleichen Sie imperative und deklarative Programmierparadigmen mit Hilfe einer Literatur-/Internetrecherche! (Optionale Aufgabe)

Algorithmen und Programmiersprachen

Begriffe formaler Sprachen:

- Ein Alphabet ist eine endliche Menge A von Zeichen und/oder Symbolen.
- Ein Wort ist eine endliche Folge von Elementen eines Alphabets. A* bezeichnet die Menge aller Wörter über einem Alphabet A.
- Eine formale Sprache ist eine Teilmenge S von A*.
- Die Syntax (Lehre vom Satzbau) gibt an, welche Wörter zu einer formalen Sprache gehören und in welcher Weise Programmtexte gebildet werden dürfen. Dazu werden Regeln formuliert, die z.B. mit Hilfe der Backus-Naur-Form – BNF oder mittels Syntaxdiagrammen beschrieben werden.
- Die Semantik einer formalen Sprache legt ihre Bedeutung fest. Dabei wird jedes Wort der Sprache durch eine sog. Semantikfunktion hinsichtlich seiner Wirkung interpretiert.

Algorithmen und Programmiersprachen

Lexikalische Regeln (zulässige Wörter einer Prg.-Sprache – hier Pascal):

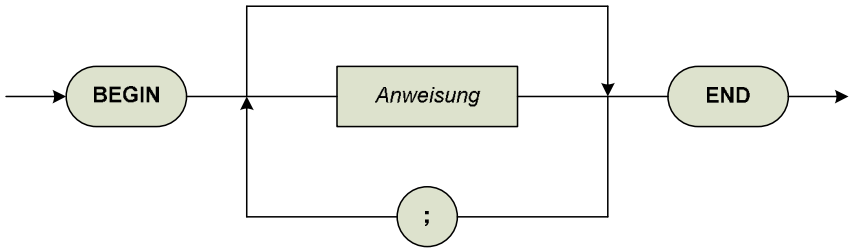
- Schlüsselworte: `PROGRAM, VAR, BEGIN, END, IF, THAN, WHILE,…`
- Bezeichner: Selbst gewählte Namen für Konstanten, Variablen, …
- Spezialsymbole: `+, -, *, /, =, <, >, <=, >=, <>, [, ], :=,…`
- Datentypen:
  - Zahlenkonstanten: `Integer, Real, Hex, Boolean, …`
  - Character & Stringkonstanten: eingeschlossen in `„c“` oder `„test“`
- Kommentare: `{ ….}` bzw. `(* ….*)`

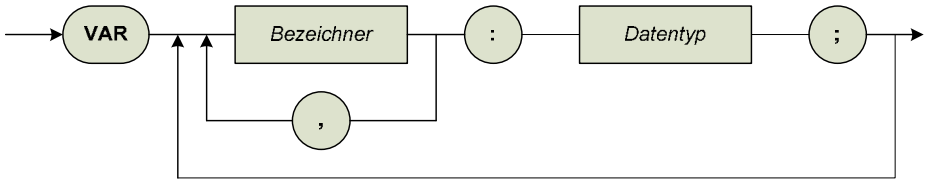
Quelle: Gumm, H. P.; Sommer, M.: Einführung in die Informatik, 8. Auflage. München, Oldenbourg Verlag, 2009

Algorithmen und Programmiersprachen

Syntaxdiagramme:

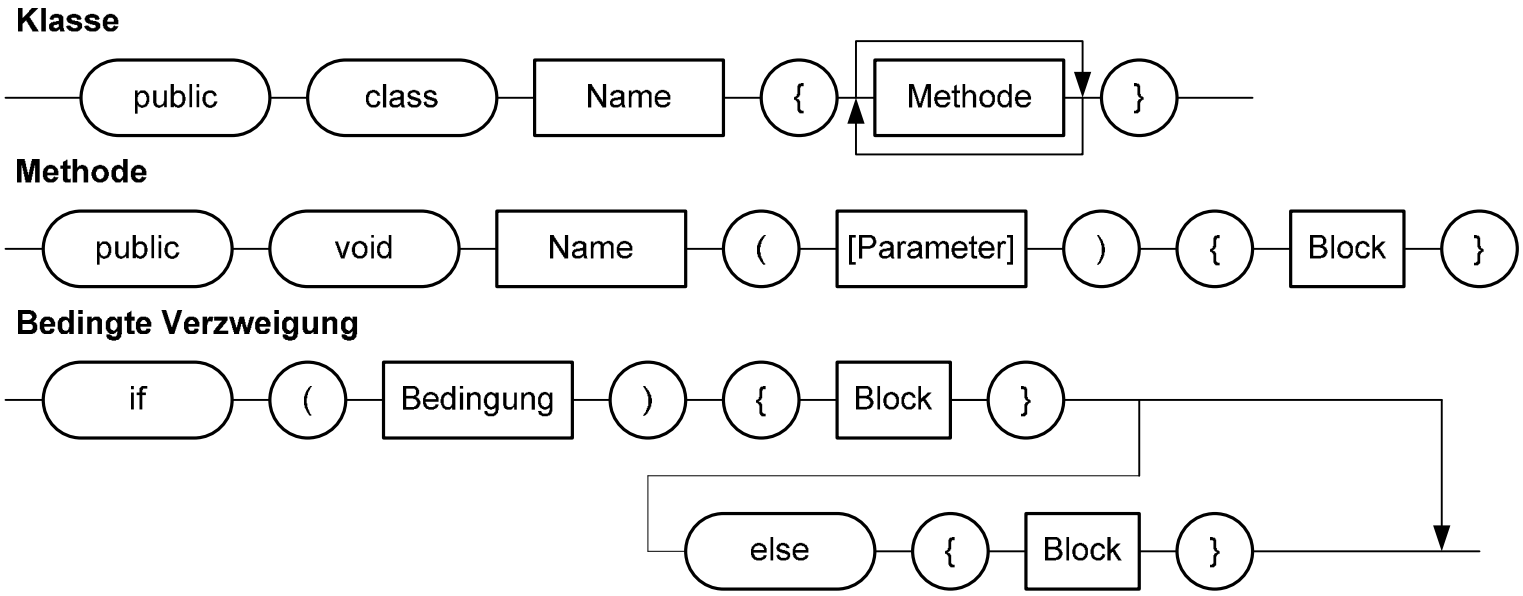
- Grafische Darstellung der Syntax einer Programmiersprache
- Nicht-Terminalsymbole
  - Nonterminale
  - Erklärt durch eigene Syntaxdiagramme
- Terminalsymbole
  - Terminale
  - Erscheinen wörtlich im Programmtext
- Es existiert genau ein Eingang und ein Ausgang





Quelle: Gumm, H. P.; Sommer, M.: Einführung in die Informatik, 8. Auflage. München, Oldenbourg Verlag, 2009

Algorithmen und Programmiersprachen



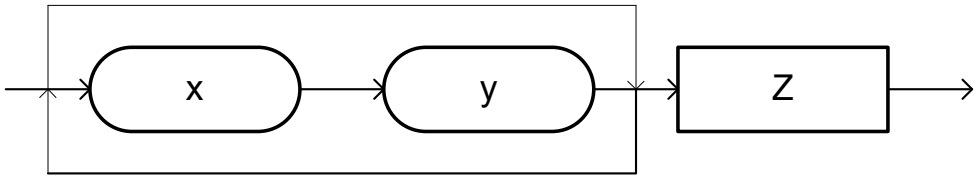
Bem.: Blöcke werden in Java mit einer öffnenden und schließenden geschweiften Klammer begrenzt. Ein Block kann Leeranweisungen, Ein- und Ausgabeanweisungen, Zuweisungen und Berechnungen, aber auch `if/switch` Anweisungen oder auch `for/while/do while` Anweisungen oder auch Methodenaufrufe enthalten.

Übung Syntax und Semantik

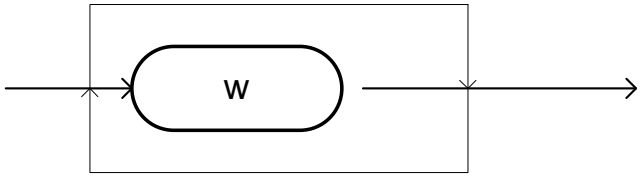
Was ist syntaktisch korrekt, entsprechend der Syntaxdiagramme?

- `xyxyxy`
- `xxxw`
- `xyxyxywww`
- `wwwwwww`
- `xyxyx`
- `wxyxyxy`
- `xyZwww`
- `AxyZ`

Block A



Block Z



5 min. Übung und gemeinsame Auswertung

Algorithmen und Programmiersprachen

Syntax von Programmiersprachen:

Programmiersprachen sind umfangreiche formale Sprachen.
Programme sind dementsprechend syntaktisch korrekte Sätze.

```
int ggt(int a, int b) {
    if (a == b){
            return a;
    }
    else if (a < b){
            return ggt(a, b - a);
    }
    else{
            return ggt(a - b, b);
    }
}
```

Übung Syntax und Semantik

- Nennen Sie Ihnen bekannte Regeln zur Syntax von Texten, die in deutscher, englischer oder russischer Sprache abgefasst sind?
- Identifizieren Sie im folgenden potentielle Syntaxregeln:
  - Bezeichner (z.B. Variablen) dürfen nicht mit einer Ziffer beginnen.
  - Programme müssen ein korrektes Ergebnis liefern.
  - Jede Programmzeile ist mit einem Fragezeichen abzuschließen.
  - Öffnende und schließende geschweifte Klammern sind immer paarweise bzw. „balanciert“ einzusetzen.
  - Variablen müssen vor ihrer Verwendung deklariert werden.
  - Programme müssen nach endlicher Zeit terminieren.
  - Wertebereiche eines Datentyps dürfen nicht überschritten werden.

Algorithmen und Programmiersprachen

Implementierung von Programmiersprachen (I)

- Computer verwenden prozessoreigene Befehlssätze. Diese können in der maschinennahen Programmierung zur Umsetzung von Algorithmen verwendet werden.
- **Jedoch**:
  - Komplexere Programme sind unübersichtlich und schwer lesbar.
- **Lösung**:
  - Verwendung höherer Programmiersprachen → Abstraktion
  - Übersetzung der Programme in semantikerhaltene Maschinensprache durch einen Übersetzer (Compiler)
  - Interpretation, d.h. unmittelbare Ausführung der Programme durch einen Interpretierer (Interpreter).

Algorithmen und Programmiersprachen

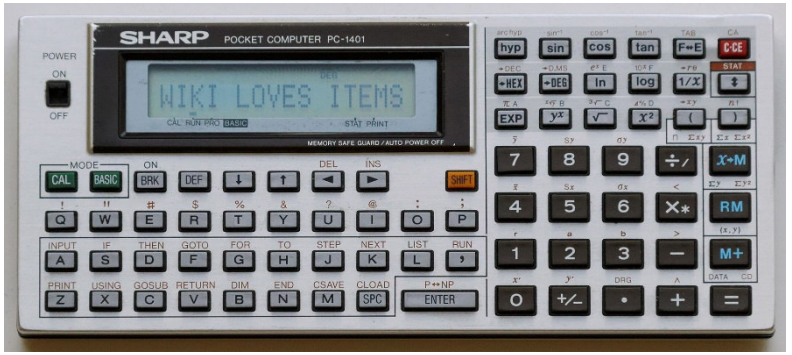
Implementierung von Programmiersprachen (II)

- Compiler und Interpreter sind selbst Programme, die Programme in der entsprechenden Programmiersprache als Eingabe akzeptieren.
- **Compiler**
  - erzeugen als Ausgabe ein entsprechendes Programm in der jeweiligen Maschinensprache.
    Forderung: Die Übersetzung ist semantikerhaltend, d.h. das erzeugte Maschinenprogramm hat die gleiche Wirkung wie das Quellprogramm (siehe auch Fachgebiet Compilerbau)
- **Interpreter**
  - führen das eingegebene Quellprogramm unmittelbar aus, d.h. sie stoßen die Aktionen an, die durch das Programm vorgegeben sind.

Programmiersprache BASIC

- Basic hat aufgrund der Verfügbarkeit entsprechender Interpreter im ROM eine starke Verbreitung erfahren. Wegen anfänglich fehlender Strukturierungsmöglichkeiten wurden Konzepte der prozeduralen- und objektorientierten Programmierung sukzessive aufgenommen.
- *Beispiel*:

```
10 INPUT M
20 INPUT N
30 IF M = N THEN GOTO 90
40 IF M <= N THEN GOTO 70
50 LET M = M - N
60 GOTO 30
70 LET N = N - M
80 GOTO 30
90 PRINT M
100 END
```



Blick zurück: https://pockemul.com/OL/pockemul.html

Quelle der Abb: https://de.wikipedia.org/wiki/Sharp_PC-1401#/media/Datei:Sharp_PC-1401_high-res.jpg

Programmiersprache Pascal

- Programme setzen sich aus *Folgen von Anweisungen* zusammen. Elementar ist die *Wertzuweisung*. Der *Zustand* im Speicher und des Ablaufs gibt den Stand eines Programms an. *Iteration* ist wichtige Kontrollstruktur, allerdings existieren auch OO-Erweiterungen.
- *Beispiel*:

```
PROGRAM ggT;
CONST M = 60; N = 30;
VAR x,y,z : Integer;
BEGIN
   x:= M; y:= N;
   WHILE x <> y DO
        IF x > y       THEN x:= x - y
                       ELSE y:= y - x
   z:= x
END.
```

Programmiersprache Java

- Programme sind Familien von Klassen, die Daten als Attribute und Algorithmen als Methoden zusammenfassen (Prinzip der Kapselung). Strukturierung erfolgt durch Klassenhierarchien, das Prinzip der Vererbung (von Attributen und Methoden) und Beziehungen.
- *Beispiel*:

```
public class MathGGT {
        public static void main(String[] args) {
        int x = 30; int y = 80; int z;
        while (x != y){
                if (x > y) x = x-y;
                else y = y-x;
        }
        z = x;
        System.out.println(z);
        }
}
```

Übung Algorithmen

- Welche Möglichkeiten gibt es, sich wiederholende Programmteile zu programmieren? Welche Paradigmen der Programmierung sind auf welche dieser Möglichkeiten ausgerichtet?
- Nennen Sie Ihnen bekannte Vor- und Nachteile von Compiler- bzw. Interpretersprachen.
- Die Programmiersprache Java ist eine Kombination von Compiler- (Quellcode) und Interpretersprache (Bytecode). Zur Steigerung der Ausführungsgeschwindigkeit werden anstatt des Interpreters Just-in-Time-Compiler eingesetzt. Machen Sie Vorschläge, wie ein solcher Compiler arbeiten könnte!