Einführung in die IT

Teil3:
Algorithmen, Datenstrukturen, XML

Bernd Schöner

DHBW Mannheim

Algorithmus

- Ein **Algorithmus** ist eine eindeutige Handlungsvorschrift zur Lösung eines Problems oder einer Klasse von Problemen.
- Algorithmen bestehen aus endlich vielen, wohldefinierten Einzelschritten. Damit können sie zur Ausführung in ein Computerprogramm implementiert, aber auch in menschlicher Sprache formuliert werden.
- Bei der Problemlösung wird eine bestimmte Eingabe in eine bestimmte Ausgabe überführt nach dem Schema: Eingabe-Verarbeitung-Ausgabe (EVA).
- Mit Algorithmen kann man eine Klasse von Problemen lösen (z.B. Suchen, Sortieren, Optimieren..).
- Die Lösung eines bestimmten Problems erfolgt durch Parametrierung - Beispiel.: Schwellenwert bei einer Gruppierung.
- Der Algorithmus muss so allgemein formuliert sein, dass er auf die Problemklasse anwendbar ist.
- Ein praxistauglicher Algorithmus muss effektiv und effizient.

Algorithmus, Eigenschaften

Eigenschaften eines Algorithmus:

1. Das Verfahren muss in einem endlichen Text eindeutig beschreibbar sein (Finitheit).
2. Jeder Schritt des Verfahrens muss tatsächlich ausführbar sein (Ausführbarkeit).
3. Das Verfahren darf zu jedem Zeitpunkt nur endlich viel Speicherplatz benötigen (Dynamische Finitheit).
4. Das Verfahren darf nur endlich viele Schritte benötigen (Terminierung). Für manche Abläufe ist ein nicht-terminierendes Verhalten gewünscht: Z. B. Steuerungssysteme, Betriebssysteme und Programme, die auf Interaktion mit dem Benutzer aufbauen.

In praktischen Bereichen wird der Begriff Algorithmus oft auf die folgenden Eigenschaften eingeschränkt:

1. Der Algorithmus muss bei denselben Voraussetzungen das gleiche Ergebnis liefern (Determiniertheit).
2. Die nächste anzuwendende Regel im Verfahren ist zu jedem Zeitpunkt eindeutig definiert (Determinismus).

Algorithmus, weitere Charakteristika

- Klassische Einzelschritte eines Algorithmus sind Rechenanweisungen und Kontrollanweisungen, wie Sequenzen, Fallunterscheidungen (Selektion), Wiederholungen (Iterationen) und Rekursionen. Diese Konzepte finden wir in Programmiersprachen wieder, mit deren Hilfe wir Programme erstellen.

- Beschreibung der funktionalen Abhängigkeit zwischen den Ein- und Ausgabedaten. Entwicklung eines Algorithmus, Prüfung auf Allgemeingültigkeit.

- Ein **Programm** ist eine Beschreibung eines Algorithmus und seiner von ihm bearbeiteten Daten in einer Programmiersprache.

- **Programmierung:** Beschreibung der Einzelschritte des Algorithmus durch eine Programmiersprache, Überführung in ein ausführbares Programm. Die Auswahl der Sprache und Überführung (Kompilierung) hat Einfluss auf die Effizienz.

- **Ausführung:** Durchführung der Einzelschritte des Algorithmus durch Ausführung der Instruktionen des Programms. Dabei hat das Betriebssystem bzw. die zugrunde liegende Hardware einen Einfluß auf die Effizienz (CPU vs. GPU).

Algorithmus, Komplexität

- Die Komplexität eines Algorithmus oder einer berechenbaren Funktion wird definiert durch den Aufwand an Betriebsmitteln, der für seine Durchführung notwendig ist. Wesentliche Betriebsmittel sind Laufzeit, Anzahl der Rechenschritte, Speicherbedarf und Geräte.

- Typische Beispiele für in der Informatik auftretende Komplexitäten (O: Komplexitätsklasse oder Aufwandsmaß):

Bezeichnung	**Komplexität**	**Bewertung**	**Beispiele**
Konstante Komplexität	O(1)	optimal	Hashing, PUSH/POP (Stack)
Logarithmische Komplexität	O(log n)	extrem gut	Binäre Suche in sortierter Liste
Lineare Komplexität O(n)		sehr gut	Lineare Suche in unsortierter Liste
Überlineare Komplexität	O(n log n)	gut	gute Sortierverfahren Mergesort, Quicksort (im Mittel)
Quadratische Komplexität	$O(n^2)$	schlecht	schlechte Sortierverfahren, z. B. Bubblesort, Quicksort (worst case)
Kubische Komplexität	$O(n^3)$	schlecht	Matrix-Multiplikation
Exponentielle Komplexität	$O(a^n)$	extrem schlecht	Travelling-Salesman
Faktorielle Komplexität	O(n!)	noch schlimmer	Travelling-Salesman (brute-force)

Daten/Datentypen

- Ein Algorithmus arbeitet mit Eingabedaten und produziert Ausgabedaten.
- Die Datenumwandlung ist eng mit den Datentypen und den zugehörigen Wertebereichen verknüpft, z.B. Zahlen, Buchstaben, Listen, Bäume und Datenstrukturen.
- Durch die Datentypen ist bestimmt, welche Operationen auf den Daten des entsprechenden Typs möglich und erlaubt sind.
- Die Daten werden als Objekte betrachtet. Ein Objekt hat eine Grösse, einen Bezeichner (Identifikator) und einen Wert.
- Datentyp ist ein grundlegender Begriff der Informatik. Es geht um die Charakterisierung der Objekte durch die zulässigen Operationen. Objekte und Operationen werden zu einer Einheit zusammengefasst.
- Rechnerinterne Darstellung der Objekte und die Implementierung der Operationen wird hier nicht betrachtet.
- Damit sprechen wir von **abstrakten Datentypen**.

Abstrakte Datentypen

- J. Guttag hat Mitte der 1970er Jahre das Konzept der abstrakten Datentypen (ADT) definiert.
- Ein abstrakter Datentyp macht seine Variablen (Daten) nicht mehr öffentlich per Zuweisung änderbar, sondern er bietet nur noch Zugriffsfunktionen (Operationen) an.
- Da der Zugriff nur über die festgelegten Operationen erfolgt, sind die Daten nach außen gekapselt.
- Die gekapselten Variablen gehören zur Innensicht und können daher flexibel umprogrammiert werden.
- Ein ADT beschreibt, was die Operationen tun (Semantik), aber noch nicht, wie sie es tun sollen (Implementierung).
- Modernere Programmiersprachen unterstützen gezielt die Erstellung von ADTs. Zum Beispiel unterstützt Java ADTs durch Klassen, abstrakte Klassen und Interfaces.

Grundlegende Datenstrukturen

- Eine Datenstruktur ist ein Objekt, welches zur Speicherung und Organisation von Daten dient. Die Daten werden in einer bestimmten Art und Weise angeordnet und verknüpft, um den Zugriff auf sie und ihre Verwaltung effizient zu ermöglichen.
- Man muss erst Objekte (Daten) definieren, bevor man Algorithmen auf sie anwenden kann. Die optimale Verbindung von Datenstrukturen mit darauf abgestimmten Algorithmen ist eine wichtige Voraussetzung für effizientes Programmieren.
- Die Festlegung (Definition) von Datenstrukturen erfolgt im Allgemeinen durch eine exakte Beschreibung (Spezifikation) zur Datenhaltung und der dazu nötigen Operationen.
- **Beispiel**: Abbildung von materiellen oder immateriellen Dinge oder Zuständen:
  - Person (Vor-/Nachame, Alter, Wohnort..)
  - Artikel (Bezeichnung, Preis, Mengeneinheit..)
  - Vertrag (Konditionen, Abschlussdatum, Laufzeit..)

Datentypen/1

- Unterscheidung zwischen:
  - Elementare Datentypen
  - Dynamische Datentypen
  - Zusammengesetzte Datentypen

- ***Elementare Datentypen:***
  - Zahlen, wie INT, FLOAT
  - BOOL (logische Werte), wahr oder falsch
  - Buchstaben und andere Zeichen
  - Aufzählungen

Datentypen/2

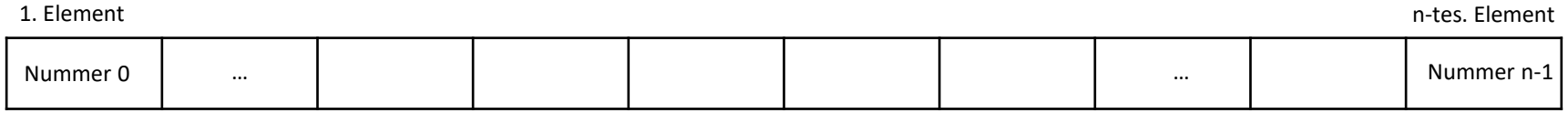
- ***Dynamische Datentypen:***
  - Zeiger (Pointer) sind Referenzen auf beliebige andere Datentypen
- ***Zusammengesetzte Datentypen***, oder Datenstrukturen, welche aus einfacheren Datentypen bestehen:
  - Feld, Vektor oder Array [n]
  - Zeichenkette (String) mit fester oder variabler Länge
  - binäre Zeichenketten variabler Länge (BLOB)
  - Satz, Struktur, eine Folge verschiedener Komponenten, welche verschiedene Datentypen haben können

Zeiger

- Zeiger sind Konstanten oder Variablen, die als Werte typisierteAdressen enthalten.
- Zeiger verweisen demnach auf Speicherplätze und geben gleichzeitig den Typ der Daten an, die dort gespeichert werden können.
- Bei der Deklaration müssen Zeiger durch den Indirektionsoperator * gekennzeichnet werden.
  - Typ *zeigername;
- Zeiger werden unter anderem dazu verwendet, dynamischen Speicher zu verwalten. So werden bestimmte Datenstrukturen, zum Beispiel verkettete Listen, in der Regel mit Hilfe von Zeigern implementiert.
- Eine der wichtigsten Anwendung ist, diese als Parameter in Funktionen zu übergeben
  - Dadurch wird vermieden, dass platzraubende Kopien größerer Objekte, beispielsweise Strukturen oder Arrays, als Parameter übergeben werden müssen.
  - Es werden lediglich Zeiger als Referenzen auf Daten übergeben.
  - Dies führt zu einem Geschwindigkeitsvorteil und zu einer effizienteren Speicherausnutzung.

Array

- Ein Feld (engl. Array, manchmal auch Tabelle, Vektor, Reihe, Reihung, Datenfeld, Aufstellung, Bereich genannt) ist in der Informatik eine Datenstruktur-Variante, mit deren Verwendung „viele gleichartig strukturierte Daten […] verarbeitet werden sollen“.
- Auch die einzelnen Elemente eines Arrays werden mit unterschiedlichen Ausdrücken bezeichnet: Element, Komponente, Unterfeld, Feldelement, indizierte Variable – zum Teil ebenfalls „Feld“ oder „Datenfeld“.
- Zur Adressierung eines einzelnen Elements in einem Feld wird ein sogenannter ‚Index‘ verwendet. Bei mehrdimensionalen Feldern gibt es für jede Dimension einen Index.



- Arrays werden von den Compilern der Programmiersprachen, zum Teil auch in verschiedenen Sprachversionen, unterschiedlich umgesetzt und unterstützt (Beginn mit 0 oder 1).

- **Beispiel**: Eine Array Vereinbarung in C:

```
type Name [Grösse];          allgemeine Darstellung
int Zahlenfeld [10];         Zahlenfeld enthält 10 integer Zahlen
```

…

String

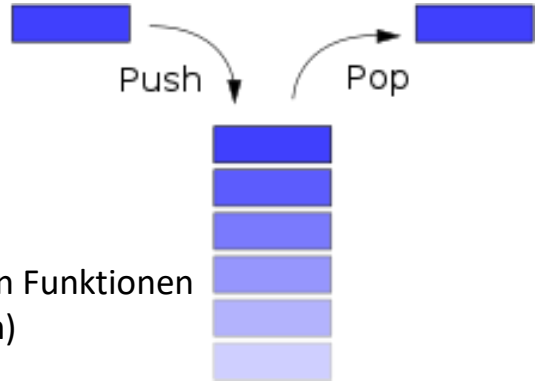
- Eine Zeichenkette, Zeichenfolge, Zeichenreihe oder ein String ist in der Informatik eine endliche Folge von Zeichen mit fester oder variabler Länge (z. B. Buchstaben, Ziffern, Sonderzeichen und Steuerzeichen) aus einem definierten Zeichensatz.
- Ein String kann auch leer sein, also kein Zeichen enthalten und die Länge 0 haben. Zeichenketten sind somit Sequenzen aus Symbolen.
- Beispiel: 'Dieser Satz ist eine Zeichenkette.'
- Repräsentation mit Abschlusszeichen:
  - In Programmiersprachen wie C werden die Zeichenketten fortlaufend im Speicher abgelegt und mit dem Nullzeichen (NUL in ASCII) abgeschlossen.
- Repräsentation mit separater Längenangabe:
  - Längenfeld
- Die Basisoperationen mit Zeichenketten, die in fast allen Programmiersprachen vorkommen, sind Länge, Kopieren, Vergleichen, Verketten, Bilden von Teilketten, Mustererkennung, Suchen von Teilketten oder einzelnen Zeichen.

…

…

Stapelspeicher

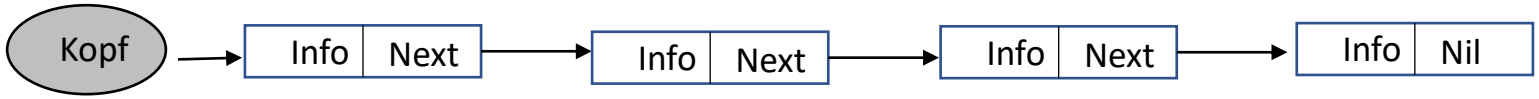
- Stapelspeicher, Kellerspeicher (kurz Stapel oder Keller, häufig auch Stack genannt) eine häufig eingesetzte dynamische Datenstruktur.
- Ein Stapel kann eine theoretisch beliebige, in der Praxis jedoch begrenzte Menge von Objekten aufnehmen. Elemente können nur oben auf den Stapel gelegt und auch nur von dort wieder gelesen werden. Elemente werden übereinander gestapelt und in umgekehrter Reihenfolge vom Stapel genommen. Dies wird auch Last-In-First-Out-Prinzip (LIFO) genannt
- Stack Operationen sind:
  - push (auch „einkellern“), legt das Objekt oben auf den Stapel.
  - pop („auskellern“), liefert das oberste Objekt und entfernt es vom Stapel.
  - peek („nachsehen“), liefert das oberste Objekt, ohne es zu entfernen (zuweilen auch top genannt, also „oben“).
  - IsEmpty: Ist der Stack leer?
  - IsFull: Ist der Stack komplett belegegt?
- Beispielanwendungsbereiche: Automatentheorie, Compiler



Vereinfachte Darstellung eines Stacks mit den Funktionen Push (drauflegen) und Pop (herunternehmen)

Listen/1

- Die verkettete Liste ist eine dynamische Datenstruktur, die eine geordnete Speicherung von Datenelementen implementiert. Die Anzahl der Objekte muss dabei nicht im Vorhinein bekannt sein und bleibt für die gesamte Lebenszeit der Liste offen.
- Grundkonstruktion: Objekte haben Referenz auf Objekt der eigenen Klasse oder Pointer
- **Einfach verkettete Listen**, einzelne Knoten, die aus den Nettodaten selbst und einem Verweis auf den Nachfolgeknoten bestehen. Im letzten Knoten ist als Nachfolgeknoten der sogenannte Null-Zeiger angegeben.



- Alle Listenelemente sind vom gleichen Datentyp (homogene Datenstruktur).
- Das erste Listenelement hat keinen Vorgänger, alle anderen haben genau einen. Das letzte Listenelement hat keinen Nachfolger, alle anderen haben genau einen Nachfolger.
- Es existiert ein als Listenkopf bezeichneter, ausgezeichneter Zeiger, der auf das erste Listenelement deutet.
- Anwendungen: Einfach verkettete Listen werden in hochdynamischen Umgebungen verwendet.
- Typische Operationen: Anlegen, Finden von Elementen, Einfügen von Elementen, Durchlaufen aller Elemente, Löschen eines Elementes.
- Die Operationen Einfügen und Löschen sind auf linearen Listen sehr effizient durchführbar, da sie sich auf das Versetzen von Zeigern beschränken.

...

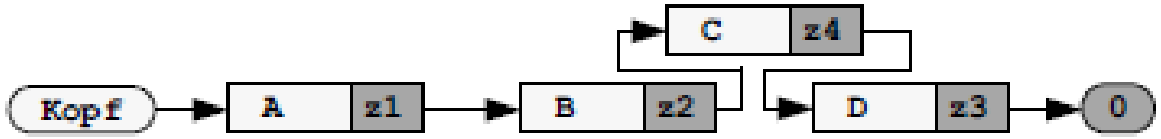
Listen/2

- **Doppelt verkettete Listen** mit drei Werten.
- Im Gegensatz zur einfach-verketteten Liste hat jedes Element, ausser den Nutzdaten, sowohl einen Zeiger auf das nachfolgende als auch auf das vorhergehende Element.
- Der Vorgänger-Zeiger des ersten und der Nachfolger-Zeiger des letzten Elementes zeigen auf den Wert NULL. Dieses besondere Element dient zum Feststellen des Anfangs und des Endes einer doppelt verketteten Liste.
- Beispiel: Doppelt verkettete Liste mit drei Werten:

...

Einfach verkettete Liste:

Einfügen des Elements C:



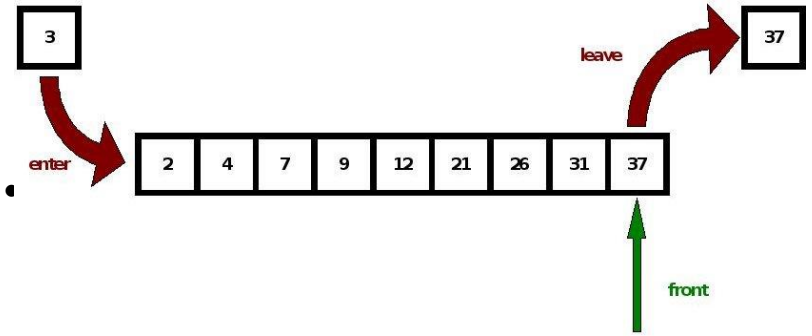
Löschen des Elements B:

Quelle: Grundkurs Informatik, H. Ernst, J. Schmidt und G. Beneken.

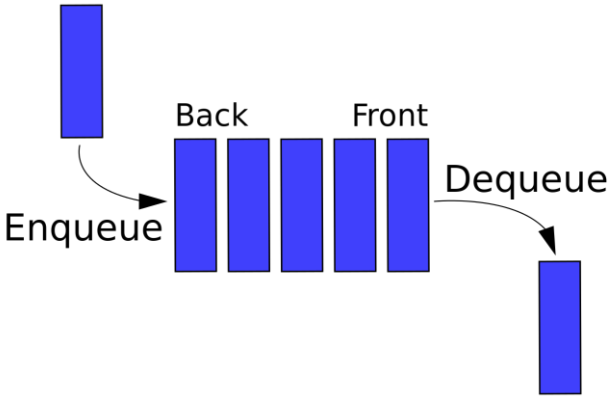
...

Warteschlangen

- Eine Warteschlange (Queue) ist ein Datenstruktur zur Zwischenspeicherung von Objekten in einer Reihenfolge, bevor diese weiterverarbeitet werden.



https://de.wikipedia.org/wiki/Warteschlange_(Datenstruktur)#/media/Datei:Queue_algorithmn.jpg



https://commons.wikimedia.org/w/index.php?curid=7586271

- Operationen bei Queues:
  - enqueue zum Hinzufügen eines Objekts und
  - dequeue zum Zurückholen und Entfernen eines Objektes bereit.
  - IsEmpty, IsFull, Peek (erstes Element abfragen)
- Es wird nach dem Prinzip First In – First Out (kurz FIFO, deutsch zuerst hinein – zuerst heraus) gearbeitet, das heißt, es wird von dequeue immer das Objekt aus der Warteschlange zurückgegeben, welches von den in der Warteschlange noch vorhandenen Objekten als erstes mit enqueue hineingelegt wurde.
- Sonderform: Ringpuffer, Schlangen werden ringförmig geschlossen.
- Anwendungsbeispiele: Drucken, Nachrichtenkommunikation

...

Datensatz oder Record

- Ein **Datensatz** ist eine Gruppe von inhaltlich zusammenhängenden (zu einem Objekt gehörenden) Datenfeldern, z. B. Name und Anschrift.
- Datensätze entsprechen einer logischen Struktur, die bei der Softwareentwicklung (z. B. im konzeptionellen Schema der Datenmodellierung) festgelegt wurde.
- Die zu Datensätzen zusammengefassten Daten werden in Datenbanken oder in Dateien gespeichert.
- **Beispiel**: Abbildung von materiellen oder immateriellen Dingen oder Zuständen:

```
struct Person {
        char  vorname[50];
        char  nachname[50];
        struct Geburtsdatum {
                int jahr;
                int monat;
                int tag;
                        };
        char  wohnort[30];
        ;
```

Datensätze oder Records - Beispiel

Vorname	Nachname	Jahr	Monat	Tag	Wohnort
Anna	Schmitt	1982	10	21	Heidelberg
Peter	Müller	2001	05	03	Mannheim
Petra	Meier	1995	01	18	Frankfurt
Stefan	Schröder	1998	11	12	Berlin
…	…	…			

Extensible Markup Language (XML)

- XML, 1998 ist eine Sprache zur Darstellung hierarchisch strukturierter Daten, die als Textdatei sowohl von Menschen als auch von Programmen lesbar sein soll.
- XML wird auch für den plattform- und implementationsunabhängigen Austausch von Daten zwischen Computersystemen eingesetzt, insbesondere über das Internet.
- XML-Dateien haben von sich aus eine definierte Grundstruktur, doch sie können zusätzlich durch Dokumenttypdefinitionen (DTD) in ihrer Form zusätzlich reglementiert werden und damit auf ihre Form geprüft werden.
- XML-Dokumente, die UTF-8 oder UTF-16 verwenden, können in allen Texteditoren, die diese Kodierungen unterstützen, angezeigt und bearbeitet werden.
- Praktisch alle Webbrowser können XML-Dokumente mit Hilfe des eingebauten XML-Parsers direkt visualisieren.
- Das Einlesen von XML-Dokumenten erfolgt über eine spezielle Programmkomponente, einen XML-Parser. Er stellt eine Programmierschnittstelle (API) zur Verfügung, über die die Anwendung auf das XML-Dokument zugreift. XML-Parser-APIs sind für viele Programmiersprachen vorhanden, z. B. Java, C, C++, C#, Python, Perl und PHP.
- Zur Erstellung von XML-Dokumenten gibt es spezielle Programme, sogenannte XML-Editoren.

XML Beispiel

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Mitglieder>
    <Mitglied Id=„42">
        <Vorname>Thomas</Vorname>
        <Nachname>Maier</Nachname>
        <Strasse>Hauptstrasse 12</Strasse>
        <PLZ>12345</PLZ>
        <Stadt>Stuttgart</Stadt>
    </Mitglied>
    <Mitglied Id=„87">
        <Vorname>Michael</Vorname>
        <Nachname>Schneider</Nachname>
        <Strasse>Bahnhofsstrasse 21</Strasse>
        <PLZ>54321</PLZ>
        <Stadt>Mannheim</Stadt>
    </Mitglied>
</Mitglieder>
```

Extensible Markup Language (XML), weitere Infos

Ein wohlgeformtes XML-Dokument hat grundsätzlich die folgende Struktur, die noch vor weiteren Regeln in Dokumenttypdefinitionen eingehalten werden müssen:

- Das Dokument kann einen Header enthalten, der über die XML-Version, Zeichensatz-Encoding und Dokumententyp aufklärt.
- XML-Elemente sind von der Form <Element>…</Element> für umschließende Elemente oder <Element/> für einzelne Elemente.
- Alle XML-Elemente dürfen durch eine beliebige Anzahl von Attributen in der Form Attribut=“Wert“ ergänzt werden.
- Die Zeichen <, >, “ , & und ’ dürfen nur in Elementen vorkommen, in allen anderen Fällen müssen diese durch <, >, " & und ' ersetzt werden.
- Das gesamte Dokument bis auf den Header muss durch ein oberstes XML-Element umschlossen sein.