Grundlagen der Informatik

2. Vorlesung

Algorithmen und deren Beschreibung

Prof. Dr.-Ing. Thomas Wiedemann
Fachgebiet Informatik / Mathematik

Überblick zur 2. Vorlesung

- Vorgehensweise zur Entwicklung von Algorithmen
- Eigenschaften von Algorithmen
- Grundelemente von Algorithmen
- Beispiele

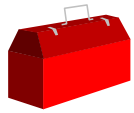
Info zur Symbolik

- Schwerpunktwissen (Definitionen/ absolute Grundlagen)
- katastrophale Auswirkungen bei Unkenntnis

Ohne Kennzeichnung

- Standardwissen - **Verständnis** und ungefähre Wiedergabe mit eigenen Worten wird in Prüfung verlangt

- zusätzliche Information
- maximal Einfluß auf die Entscheidung zwischen „Gut“ und „Sehr gut“ bzw. „Ausgezeichnet“



- Werkzeugkasten : nicht direkt prüfungsrelevant (keine direkten Fragen danach),
- sehr wichtig für die praktische Arbeit und damit indirekter Einfluß auf den Zeitfaktor bei der Prüfung

Vorgehensweise zur Entwicklung von Algorithmen

Allgemein :

- Eine Aufgabe ist meist mit verschiedenen Algorithmen lösbar !
- Es gibt gute, schlechte, elegante, trickreiche und sogar fast unverständliche Algorithmen.
- Nicht das Codeschreiben, sondern die Qualität des Algorithmus macht den Meister aus !

Aufgabe

⇩ **Denken !!!!!**

Algorithmus

Einige weit verbreitete Irrtümer (vorrangig im Managementbereich)

- Lange Algorithmen sind nicht besser als kurze ! Die Entwicklungszeit sagt ebenfalls nichts über die Qualität aus (meist sogar negativ korreliert) !
- Die Anzahl der Codezeilen sagt nichts aus über die Qualität des Programmes (großer Streitpunkt und viel Unsinn bei der Messung der Programmierleistung : bessere Programmierer schreiben weniger und 10 x effizienter !)

Hinweise für Ihre eigene Arbeit

- Denken Sie immer über mindestens eine Alternative zu Ihrer ersten Lösung nach !
- Denken Sie über den Algorithmus nach, bevor Sie den ersten Code schreiben !
- Lassen Sie den gesamten Algorithmus einmal im eigenen Kopf durchlaufen (erspart zeitaufwändige Sackgassen)

Aufgabe

⇩ **Denken**

Algorithmus

Hinweise aus eigener Erfahrung (nicht geeignet für die Prüfung)

- Gute Algorithmen entstehen nicht im Streß ! Nehmen Sie sich Zeit (in der Badewanne, bei einem Spaziergang, bei Musik) !
- Gute Algorithmen sind wie Perlen ! Man findet sie versteckt in der Mathematik, in anderen Programmen oder in der Literatur !

Der Algorithmusbegriff (Vertiefung)

Andere Algorithmusdefinitionen

- Eine endlich lange Vorschrift, bestehend aus Einzelanweisungen heißt Algorithmus.
- Ein Algorithmus ist eine Verarbeitungsvorschrift, die so präzise ist, dass sie von einem mechanisch oder elektronisch arbeitendem Gerät ausgeführt werden kann.
- Ein Algorithmus ist eine präzise, endliche Beschreibung eines allgemeinen Verfahrens unter Verwendung ausführbarer elementarer Verarbeitungsschritte.
- Ein Algorithmus ist eine detaillierte und explizite Vorschrift zur schrittweisen Lösung eines Problems. Im einzelnen bedeutet dies:
  - Die Ausführung erfolgt in diskreten Schritten.
  - Jeder Schritt besteht aus einer einfachen, offensichtlichen Grundaktion.
  - Zu jedem Zeitpunkt muss eindeutig bestimmt sein, welche Schritte als nächste auszuführen sind.

[nach Wöhner: Grundlagen der Programmierung/Algorithmierung]

Die Geschichte der Algorithmen (Hintergrundwissen)

- 300 v. Chr. : Euklids Berechnungsvorschrift für den größten gemeinsamen Teiler (Zerlegung in Primfaktoren, Vergleich ...)
- 800 n. Chr. Persischer Mathematiker alCharismi veröffentlicht Aufgabensammlung für Kaufleute, welche später in Lateinische unter dem Titel „Algorithmi“ übersetzt wird (Kunstwort aus „arithmos“ (griech. für Zahl) und dem Namen des Mathematikers
- 1574 Adam Ries schreibt Buch zu mathematischen Algorithmen
- 1614 erste Logarithmentafeln werden (30 Jahre lang!!) berechnet
- 1703 binäre Zahlensysteme und Algorithmen zu deren Berechnung werden von Leibnitz entwickelt
- 1815 Augusta Ada Lovelace (erste Programmiererin der Welt ??) entwickelt erste Programme für Rechenmaschine von Babbage
- 1931 Gödel (dt. Math.) zerstört mit seinem Unvollständigkeitssatz den Traum vieler Mathematiker, daß alle math. Sätze durch Algorithmen beweisbar sind (-> Existenz unlösbarer Problem)

[nach Saake, Sattler: Algorithmen und Datenstrukturen. 2006

Probleme der Algorithmendefinition

Probleme:

- Die Definition ist nicht exakt genug, sondern eher eine intuitive Charakterisierung des Algorithmenbegriffs.
- Die Definition zeigt den engen Zusammenhang mit **Funktionen** (Eingangsdaten -> Ausgangsdaten).
- Die Definition bindet den Algorithmus an die Lösung einer Klasse von Problemen und nicht an eine einzelne Aufgabe.
- Die Definition eines Algorithmus enthält keine Forderungen an die praktische Ausführbarkeit des Algorithmus auf einer realen Maschine (Rechner).

Berechenbarkeit von Funktionen

Schlußfolgerungen:

- Ein Algorithmus berechnet eine Funktion.
- Da die Menge der **Funktionen** überabzählbar unendlich ist, aber die Menge der Algorithmen abzählbar unendlich ist (GÖDEL), folgt das es Funktionen geben muss, denen kein Algorithmus zugeordnet werden kann. Diese sind also nicht berechenbar.

Definition:

Eine Funktion f heißt berechenbar, wenn es einen Algorithmus gibt, der für jedes Argument x den Funktionswert f(x) berechnet.

Beispiele für berechenbare und nicht berechenbare Funktionen

a) Berechne alle Primzahlen im Intervall [1,100] !
Dieses Beispiel stellt eine berechenbare Funktion dar, der Algorithmus ist relativ einfach.

b) Berechne die Funktionswerte der Funktion
f(x+1) = $2^{f(x)}$ mit f(0)=0
für 6<=x<=12 !
Dies ist auch eine berechenbare Funktion, die sich aber auf keiner realen Rechenmaschine abarbeiten lässt, da f(6) bereits eine Zahl mit 19000 Dezimalstellen ist.

c) Drucke alle ganzen Zahlen von 1 bis 10^{100} !
Auch dieses Beispiel ist berechenbar, aber nicht in endlicher Zeit auf einem realen Rechner ausführbar!

d) Berechne alle reellen Zahlen im Intervall [0,1] !
Dieses Beispiel stellt eine nicht berechenbare Funktion dar. Es kann bewiesen werden, dass solch ein Algorithmus nicht existieren kann.

Charakteristische Eigenschaften von Algorithmen

An den Beispielen ist erkennbar, dass es zweckmäßig ist, an Algorithmen solche Forderungen zu stellen, dass sie auf realen Maschinen praktisch ausführbar werden. Im folgenden werden diese Eigenschaften aufgeführt.

Endlichkeit

Ein Algorithmus muss aus einer endlichen Anzahl von Lösungsschritten bestehen und nach Abarbeitung dieser Schritte nach einer endlichen Zeit das Ende erreichen.

Eindeutigkeit

Die einzelnen Schritte eines Algorithmus und ihre Aufeinanderfolge müssen eindeutig beschrieben sein.

Allgemeinheit

Ein Algorithmus darf nicht nur die Lösung einer speziellen Aufgabe (z.B. Lösung der Gleichung $x^2 + 2x + 1=0$) , sondern muss die Lösung einer Klasse von Problemen (z.B. die Lösung aller quadratischen Gleichungen $ax^2 +bx +c =0$) beschreiben.

Determiniertheit

Die mehrmalige Anwendung des Algorithmus mit denselben Eingangsdaten muss immer wieder dieselben Ausgangsdaten liefern.

Charakteristische Eigenschaften von Algorithmen II

Effektivität

Ein Algorithmus muss real von einer Maschine ausführbar sein.

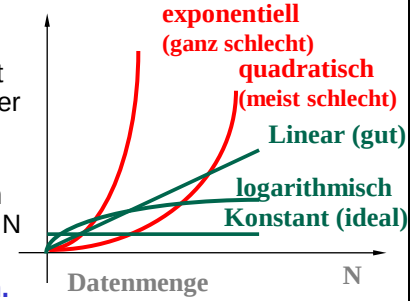
Effizienz

Ein Algorithmus muss möglichst wenig Ressourcen einer Maschine , d.h. möglichst wenig Rechenzeit und möglichst wenig Speicher und Energie in Anspruch nehmen.

In der Informatik wurden Effizienzmaße für Algorithmen, die sogenannte Zeit- und Speicherkomplexität entwickelt, mit der Algorithmen bewertet werden können. Dies ist praktisch sehr wichtig, da zur Berechnung einer Funktion meist mehrere Algorithmen angegeben werden können.

Bsp.: Komplexität von Such/Sortieralgorith. in Abhängigkeit von der Anzahl der Datensätze N

Die Informatiker suchen immer nach den besten, d.h. meist schnellsten Algorithmen.



Entwicklung von Algorithmen

- **Problem:** Die heutigen Rechner sind relativ primitiv. Sie können nur mit Zahlen und einzelnen Zeichen (Ziffern, Buchstaben, Sonderzeichen) rechnen !
- **Verfügbare Grundoperationen:** Grundrechenarten, Mathematische Funktionen (entwickelt aus Grundrechenoperationen -> Gebiet der Numerik), Vergleich und Kopieren von einzelnen Zeichen
- **Alle anderen Operationen müssen auf diese einfachen Operationen zurückgeführt werden !**

Sinnvolle Vorgehensweise (vgl. 1. Vorlesung):

1. Beschreibung des Algorithmus mit **Klartext** (auch verständlich für Laien -> Manager und Endkunden)

1b. (Optional) Darstellung mit **Hilfsmitteln** : Grafik / Unified Modeling Language, Pseudoprogrammiersprachen

2. Umsetzung in eine **Programmiersprache**

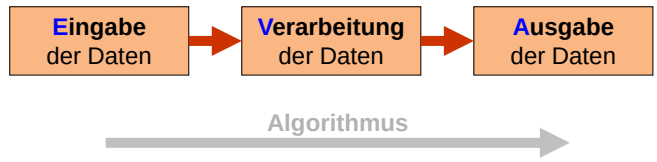
Daten und Algorithmen

Nach Niklaus Wirth (Schweizer Inf.-Prof.) ist ein Computerprogramm das Ergebnis aus **Algorithmen und Datenstrukturen**.

Daten sind die Dinge, die verarbeitet werden. Diese sind in Datenstrukturen organisiert.

Algorithmen sind die Handlungen, die Daten verarbeiten.

EVA-Prinzip der Informatik



Nach Wöhner

Algorithmusbestandteil: **Arithmetischer Ausdruck**

Ein **arithmetischer Ausdruck** ist eine Rechenvorschrift für arithmetische Werte (d.h. für Zahlen). Er entspricht einem Term der Mathematik.

Bildungsvorschriften:

- Verknüpfung von — Bsp:
  - Zahlen — -3 (ganze Zahl) 4.6 (Rationale Zahl)
  - **Variablen**, Konstanten — a x i j Sum, Pi (meist vorbelegt)
  - Funktionen — sin(a)
    sqrt(a) sqrt(2) = Wurzel ()
    abs(x+y) = Betrag ( )

  mittels
  - arithmetischer Operatoren — + - * / ( ) ^

- **Achtung: abweichende Abarbeitung/ Darstellung im Vergleich zur Mathematik**
  (orientiert an Programmiersprachen)
  - Priorität bei der Berechnung: **( )** vor **Fkt.** vor **^** vor *** /** vor **+ -**
  - Eine Informatik-Variable ist quasi ein benannter Speicherplatz und beinhaltet einen **KONKETEN WERT** !
  - deutsches Komma durch Punkt ersetzen bei Zahlen !
  - Multiplikationsoperator ***** immer schreiben, also nicht 2a !
  - meist einzeilige Schreibweise
  - nur runde Klammern **( )** zulässig für Änderung der Prioritätenreihenfolge

a **33.56**

Algorithmusbestandteil: **Logischer Ausdruck**

Ein **logischer Ausdruck** ist eine Rechenvorschrift für logische Werte (d.h. für wahr oder falsch). Logische Werte sind Steuerdaten bei der Abarbeitung eines Algorithmus.

Wertebereich: zweiwertig: *wahr/falsch w/f true/false t/f 1/0 L/0*

Bildungsvorschriften:

logischer Ausdruck:
Vergleich oder
Verknüpfung von Vergleichen
mittels logischer Operatoren

Vergleich:
Verknüpfung von zwei arithmetischen Ausdrücken
mittels eines Vergleichsoperators

logische Operatoren

- ¬ Negation
- ∧ Konjunktion, logisches ***UND***
- ∨ Disjunktion, logisches ***ODER***

Vergleichsoperatoren

- < kleiner als
- ≤ kleiner gleich
- = gleich
- > größer als
- ≥ größer gleich
- ≠ Ungleich <> !=

- Prioritäten bei der Berechnung:
  ( ) vor **arithmetischer Operator** vor **Vergleichsoperator** vor **logischer Operator**
  und ¬ vor ∧ vor ∨

Algorithmusbestandteil: Die Ergibtanweisung

Die **Ergibtanweisung** ist eine Anweisung, mit der einer **Variablen** ein Wert zugewiesen wird, der vorher durch eine Rechenvorschrift (d.h. durch einen Ausdruck) berechnet wurde.

a = b* 3 + c

Bedeutung: Der Wert der linken Variable ergibt sich aus dem Wert des rechten Ausdrucks

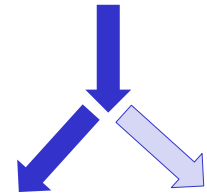
Bemerkungen:

- Die Variable links der Ergibtanweisung erhält einen neuen Wert. **Ihr alter Wert geht verloren**.
- **Die Wertzuweisung kann als echte Transportoperation von Daten aufgefasst werden.**
- **Die Wertzuweisung von rechts NACH links kann nicht getauscht werden, d.h. der mathematisch korrekte Ausdruck b*3 = a ist in der Informatik (bzw. Programmierung) NICHT korrekt !!!**
- Vor der Berechnung muss jeder Variable des Ausdrucks (steht rechts der Ergibtanweisung) einen Wert besitzen.
- Die Werte der Variablen des Ausdrucks müssen so beschaffen sein, dass alle Operationen eindeutig ausführbar sind.

Selektion / Alternative

WENN **Logischer Ausdruck**
DANN **Anweisungen**

(SONST **Anweisungen)**



… nächste Anweisungen …

Die Selektion wertet den Wahrheitswert des logischen Ausdruck aus.

Bei einem WAHREM log. Ausdruck werden alle Anweisungen im DANN-Bereich ausgeführt. Danach werden die nach dem WENN-DANN-(SONST)-Bereich folgenden Anweisungen ausgeführt.

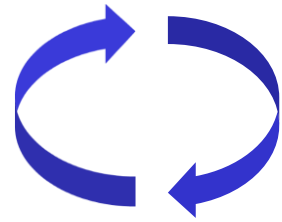
Bei einem FLASCHEM log. Ausdruck werden alle Anweisungen im SONST-Bereich ausgeführt, falls dieser existiert. Falls kein SONST-Bereich existiert wird einfach mit den nächsten Anweisungen fortgesetzt.

Als Anweisungen können ALLE möglichen Algorithmusbestandteile verwendet werden, also auch wieder Selektionen oder Schleifen.

Laufbereich (Schleife)

Der Laufbereich (Schleife)

i := 0 (1) n-1



Der **Laufbereich** ***i = 0 (1) n-1*** bedeutet, dass der **Laufvariablen** (hier ***i***) (Index) zuerst der **Anfangswert** (hier ***0***) zugewiesen wird.

Der Wert in Klammern gibt die ***Schrittweite*** (hier ***1***) an, mit der der aktuelle Wert der Laufvariablen nach jeder Eingabe erhöht wird.

Die dritte Angabe stellt den **Endwert** (hier ***n-1***) dar, bis zu welchem die Laufvariable durchlaufen wird.

Eingabeanweisung / Ausgabeanweisung

Die **Eingabeanweisung** ist eine Anweisung, mit der einer Variablen durch eine Eingabe ein beliebiger Wert zugewiesen wird.

Eingabe x , y, anzahl
Eingabe "x=" , xwert

Eingabegerät (Tastatur /Maus …)

Bedeutung: Jedem Eingabeelemente wird der eingegebene Wert (meist von der Tastatur) zugewiesen. Optional kann auch ein Text VORHER ausgegeben werden, um den Nutzer die Eingabe zu erklären (Beispiel 2).

Die **Ausgabeanweisung** ist eine Anweisung, mit der die aktuellen Werte einer Variablen und/oder Text ausgegeben wird.

Ausgabe x , y, ergeb1
Ausgabe "y=" , y

10 3.87 1.34
y= 3.87

Ausgabegerät (Bildschirm / Drucker …)

Bedeutung: Ausgabe von Text und Werten auf dem Ausgabegerät (meist Bildschirm). Der Text wird als Text ausgegeben und auch die Zahlenwerte werden als Text umgewandelt.

Struktogramme

Ein **Struktogramm** ist eine Darstellungsform für einen Algorithmus.

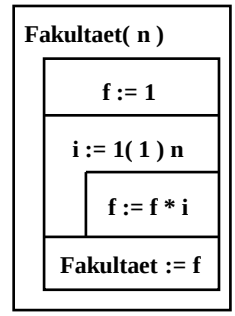
Ein **Struktogramm** stellt eine Folge von Blöcken dar, wobei jeder Block eine abgeschlossene funktionelle Bedeutung besitzt.

Eigenschaften der Blöcke:

- je Block nur 1 Eingang und 1 Ausgang
- Steuerfluss von oben nach unten
- am Eingang Datenaustausch mit Vorgängerblock (im Beispiel als Funktionsaufruf = Input)
- am Ausgang Datenaustausch mit Nachfolgerblock (Funktionsergebnis)
- Aneinanderreihen und Schachteln von Blöcken ergibt neuen Block.

Vor-und Nachteile von Struktogrammen:

- leichte Lesbarkeit
- graphische Darstellung
- hohe Transparenz Unterstützung des Vorgehens vom Allgemeinen zum Speziellen (top down)
- aufwendig bei Änderungen



Input - n als Var.

Variable f definieren und Anfangswert setzen

Laufbereich Anweisung im Laufbereich

Ergebnisrück-gabe

Beispiel 1: Berechnung der Fakultät

Aufgabe: Berechne y = f(x) = x !

Algorithmus im Klartext:

0. Prüfe auf x<2, Wenn Ja dann Programmende
1. Setze y auf x
2. Verringere x um 1
3. Multipliziere y mit x

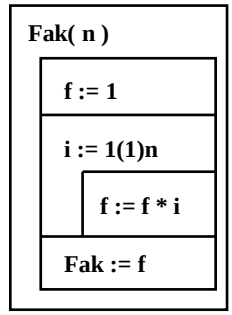
4. Beende Programm falls x kleiner/gleich 1, sonst gehe zu Schritt 2

Probleme ?

- Was passiert bei x kleiner 1 ?
- Was passiert bei x gleich 10000 ?

Beispiel 1: Berechnung der Fakultät als Grafik

mit Laufbereichszyklus



Rekursive Variante mit einfacher Auswahl

Die Besonderheit bei rekursiven Alg. besteht darin, das im Algorithmus die Funktion selbst wieder aufgerufen wird.