Allgemeines zu:

I) Grundlagen der Programmierung

VV-Nr.: 10890-01

II) Erweiterte Grundlagen der Programmierung

VV-Nr.: 45398-01

Webseiten zur Vorlesung

https://dmi.unibas.ch/de/studium/computer-science-informatik/aktuelles-semester-hs/vorlesung-grundlagen-der-programmierung/

->Grundlagen der Programmierung

- hier gibt es alle Folien zur Vorlesung
- aber auch sonstige aktuelle Informationen

-> Erweiterte Grundlagen der Programmierung

- Übungsblätter
- aber auch sonstige aktuelle Informationen

Lernziele

A) Grundlagen der Programmierung

Die Programmiersprache Java kennen.
Kleine Programme lesen und verstehen können und kurze Programmteile selbst schreiben und entwickeln können.

B) Erweiterte Grundlagen der Programmierung

Ganze Programme in Java selbst schreiben können und Erfahrung in der Umsetzung von Problemstellungen zu lauffähigen Programmen sammeln.

Überblick zur Vorlesung

Inhalt:

- Anweisungen und Datenstrukturen
- Objektorientiertes Programmieren
- Programmentwurf
- Rekursion
- Fehlerbehandlung
- Generics
- Verwenden von Java-API Bibliotheksfunktionen

Durchführung der Übungen

A) Grundlagen der Programmierung

Alle Übungen werden über ein Web-Interface in einen Web-Browser durchgeführt. Die Auswertung und die Korrekturhilfen werden automatisch erstellt.
Die vereinbarten Termine bei Tutoren können zu Fragen und Hilfestellungen genutzt werden

B) Erweiterte Grundlagen der Programmierung

Der vollständiger Programmkode muss auf einen Server geladen werden und dann einem Tutor vorgeführt werden.

Detaillierte Anweisungen finden sie auf den Webseiten zu den Kursen.

Literatur

Folien zur Vorlesung werden im Web abgelegt

Folien allein sind NICHT ausreichend!!

Bücher:

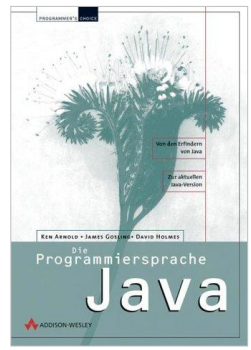
1) Sprechen sie Java,
Hanspeter Mössenböck
dpunkt Verlag etwa sFr 48,-

5. Auflage

(2.Auflage)

Das Buch der Entwickler

Arnold, K., Gosling, J. und Holmes, D. (2005). *The Java Programming Language*. Addison-Wesley. Vierte (dritte) Auflage.



Die originale Referenz, die auch einen kompakten Überblick und ein Nachschlagen während des Programmierens ermöglicht.

Setzt aber oft schon Standardwissen der Informatik voraus.

Gegen Ende der Vorlesung sehr geeignet!!

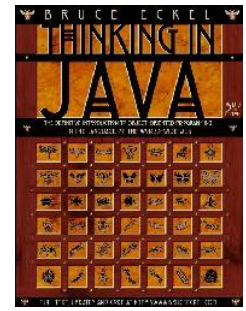
Weitere Literaturempfehlungen

Eine weitere Literaturempfehlung:

"Thinking in Java" *von Bruce Eckel*

kostenlos im Internet als pdf-Datei

http://www.mindview.net/Books/TIJ/



Einschub:
Ein erstes Java Programm.

Allgemeines zu Java

Java ist eine objektorientierte Programmiersprache in der Tradition von C und C++.

Die Java-Syntax lehnt sich an C an, nimmt aber einige deutliche Bereinigungen vor

- z. B. Abwesenheit von `goto` und dem Datentyp Zeiger (pointer).

Die Erweiterung um Objekte geschieht ähnlich wie bei C++

- Allerdings gibt es doch einige Unterschiede:
  - In erster Näherung könnte man sagen, dass Java aus der riesigen Vielfalt der C++ Konstrukte eine sinnvolle Auswahl trifft und gleichzeitig einiges umstellt und bereinigt.

„Home Page“ von Java

Java wurde bei der Firma SUN Microsystems entwickelt

- Von James Gosling
  - Ursprünglich für die Programmierung von in Haushaltselektronik eingebetteten Prozessoren

Die zentrale Internet Site der Firma Oracle zum Thema Java ist

- http://www.oracle.com/technetwork/java/index.html

Besonders empfehlenswert (und sehr empfohlen zur Ergänzung des Selbststudiums):

- Das „offizielle“ Java-Tutorial:

  http://docs.oracle.com/javase/tutorial/

Pflicht für die Teilnehmer der Erweiterten Grundlagen der Programmierung

Installation der Entwicklungsumgebung

Die Standard-Entwicklungsumgebung von SUN (z. Z. Version 1.8) kann auch direkt im Internet unter der folgenden URL heruntergeladen werden

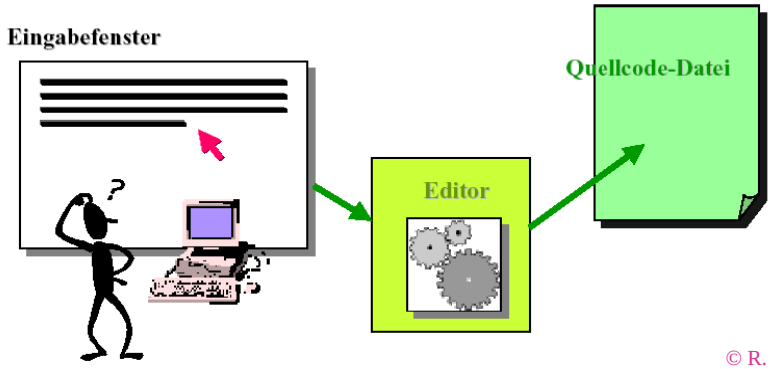
- http://www.oracle.com/technetwork/java/index.html
- Das Java Tutorial von Campione et al. (2001) enthält detaillierte Anleitungen zur Installation des SDK auf UNIX / LINUX, Apple Macintosh und Microsoft Windows Plattformen

Hilfe zur Installation im Netz: (muss Ende 2. Woche laufen)

- http://www.torsten-horn.de/techdocs/java-install.htm
- http://de.wikibooks.org/wiki/Java_Standard:_Einrichten_der_Programmierumgebung

Programmentwicklung : Editieren

Der „Quellcode“ eines Programms kann mit jedem beliebigen Editor erstellt werden.



Editoren

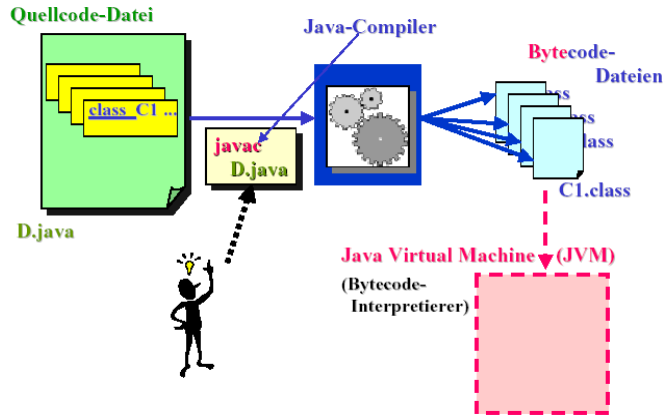
Programme zum „Schreiben“ und „Verändern“ (Editieren) von Text-Dateien heißen Editoren.

Quellcode-Dateien von Java-Programmen sind Text-Dateien

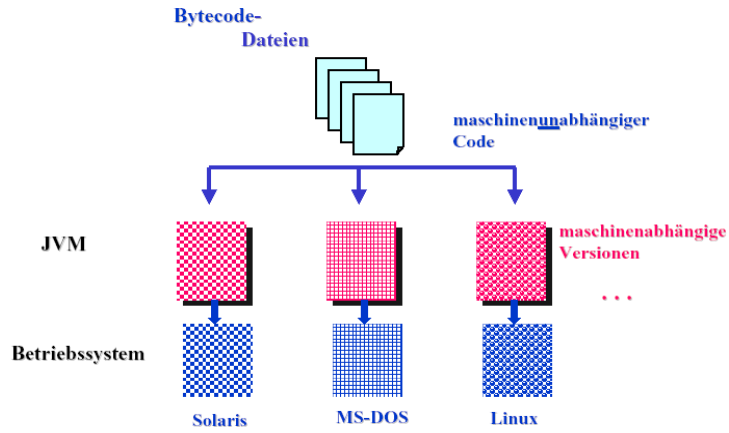
Benutzen Sie Ihren „Lieblingseditor“

- Windows: Notepad++, Geany, Jedit...
- Linux: vi, VIM, (EMACS), Geany, ...
- Max OSX: TextWrangler, ….

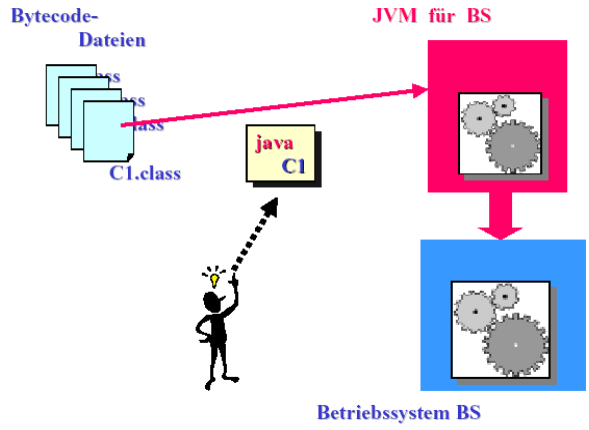
Programmiervorgang für Java



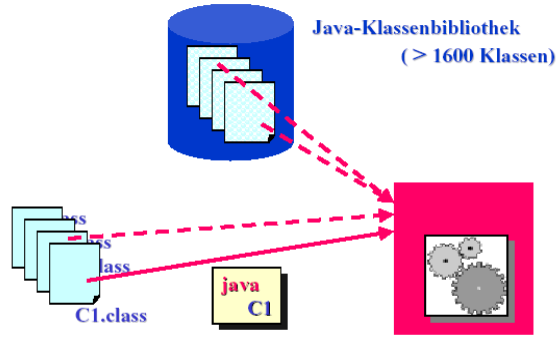
Java: Plattformunabhängiges Compilieren



Programmiervorgang für Java



Programmiervorgang für Java



automatisches Nachladen von eigenen oder vordefinierten Klassen "bei Bedarf" (d.h. wenn aufgerufen)

Java Bibliotheken

Zu Java gehört außerdem eine Sammlung von standardisierten Bibliotheken, die häufige Programmieraufgaben stark erleichtern.

Beipiele solcher Bibliotheken sind

- java.applet für Programme, die in Web-Browsern laufen sollen
- java.awt für das Erzeugen graphischer Benutzeroberflächen
- java.math für Arithmetik auf beliebig langen Zahlen (wichtig z. B. für die Kryptographie),
- java.net zum Betreiben von Verbindungen über das Internet
- java.rmi zum Aufrufen von Methoden auf entfernten Rechnern
- ...

Grundstruktur von Java- Programmen

```
class ProgramName {
        public static void main (String[] arg) {
                ... // Deklarationen
                ... // Anweisungen
        }
}
```

Text muß in einer Datei namens *ProgramName. java* stehen

Beispiel

Text steht in Datei *MyfirstProgram. java*

```
class MyFirstProgram{
    public static void main (String[] arg) {
            System.out.println( "Hallo World!");
    }
}
```

Übersetzen und Ausführen mit JDK

```
class MyFirstProgram{
    public static void main (String[] arg) {
        System.out.println( "Hallo World!");
    }
}
```

Text steht in Datei
MyfirstProgram. java

Übersetzen

C:\ cd MySamples
C:\ javac MyFirstProgram. java

wechselt ins Verzeichnis mit der Quelldatei
erzeugt Datei *MyFirstProgram.class*

Ausführen

C:\ java MyFirstProgram
Hallo World!

ruft main- Methode der Klasse
MyFirstProgram auf

Pragmatisches zur Programmierung in Java

- **Jedes Programm wird in (mindestens) einer Datei abgelegt, deren Dateiname die Endung .java aufweisen muss.**

  z.B.: **Helloworld.java**

- **Vor der ersten Ausführung eines Programms muss das Programm in eine interne Form in einer internen, maschinennahen Sprache übersetzt werden. Die Übersetzung geschieht durch Aufruf des Java-Compilers unter Angabe des Dateinamens:**

  javac Helloworld.java

- **Der Java-Compiler generiert aus jeder Klassendeklaration in der übersetzten Datei eine eigene Datei mit internem Code, die den Namen der jeweiligen Klasse mit Endung .class trägt - also im Beispiel nur eine Datei**

  Helloworld.class

Pragmatisches zur Programmierung in Java

- Nach dem Übersetzen kann dann die Datei mit internem Code zu einer der aufrufbaren Klassen gestartet werden:

```
java Helloworld
```

- Dadurch wird das Java-Laufzeitsystem zum Aufruf der main-Methode dieser Klasse aufgefordert. Alle weiteren Aktivitäten des Programms hängen dann vom Inhalt dieser Methode ab (Objekterzeugung, Aufruf anderer Methoden ggf. anderer Klassen)
- Etwas gewöhnungsbedürftig:

Compiler	`javac Helloworld.java`	mit Endung
Laufzeitsystem	`java Helloworld`	ohne Endung

Pragmatisches zur Programmierung in Java

In jeder aufrufbaren Klasse muss die Methode main mit demselben, fest vorgegebenen Methodenkopf deklariert werden:

```
public static void main (String[ ] args)
```

- überall sichtbar
- Klassen-methode
- ohne Resultat
- (String[ ] args):
  - obligatorischer Eingabeparameter (niemals weglassen).
  - Array von Zeichenreihen
  - Dient zur Übermittelung von Eingaben beim Aufruf der Klasse «von aussen».
  - Kann innerhalb von main ignoriert werden

Beispiel: Kommandozeilenparameter

```
class Program2{
    public static void main (String[] arg) {
        System.out.println(arg[0] + "_" + arg[1]);
    }
}
```

Concatenation von Zeichenketten

Ein Aufruf:

java Program2 Hallo World

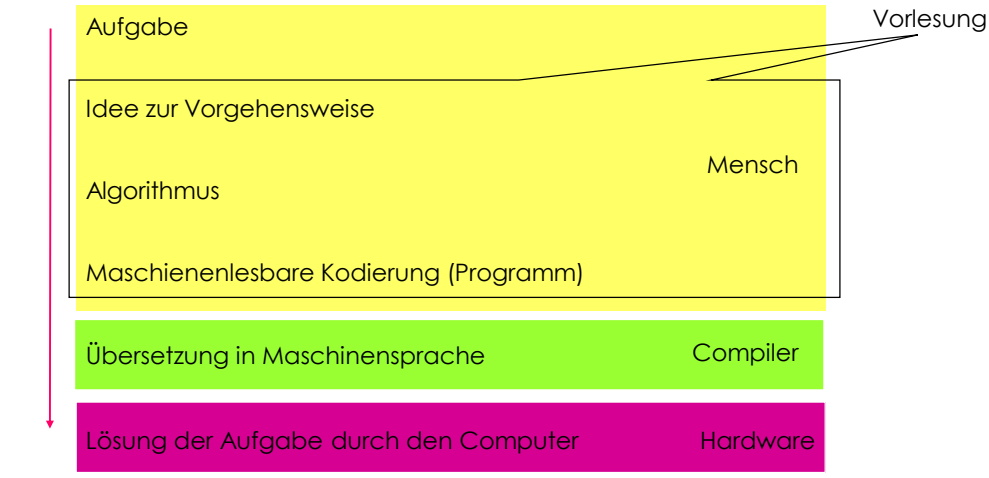
erzeugt dann

Hallo_World

Grundlagen der Programmierung: Programmieren in Java

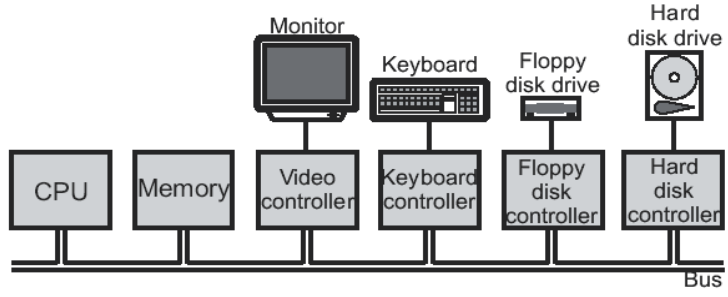
Was heisst Programmieren?

Exaktes Instruieren eines Computers, eine bestimmte Aufgabe zu lösen.



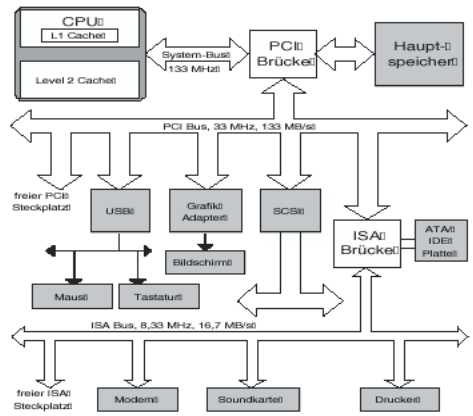
Organisation der Hardware

Architektur eines einfachen Computersystems mit Bus



System-Architektur der Hardware

Architektur eines PC Systems mit mehreren Bussen an Brücken



Software wird unterschieden in

Anwendersoftware erlaubt die Lösung allgemeinster Aufgabenstellungen.

- z. B. Textverarbeitung, Tabellenkalkulation, Bildbearbeitung, Buchhaltung, Produktionsplanung, Lohn und Gehaltsabrechnung, Spiele...

Systemsoftware hilft beim Betrieb des Rechners und bei der Konstruktion der Anwendersoftware.

- Systemsoftware umfasst neben Datenbanksystemen, Übersetzern (compiler) etc. in jedem Fall das Betriebssystem.

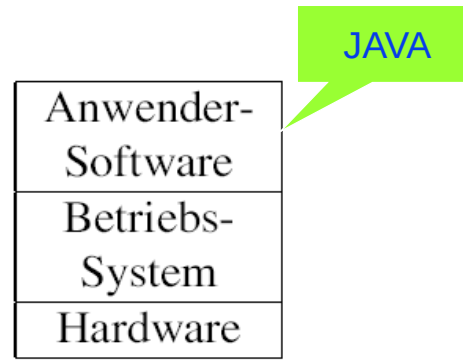
Betriebssystem

Das **Betriebssystem** (operating system) isoliert die Anwendersoftware von der Hardware: das Betriebssystem kommuniziert mit der Hardware und die Anwendersoftware auf dem Betriebssystem.

Das Betriebssystem verwaltet die Ressourcen der Hardware (wie z. B. Geräte, Speicher und Rechenzeit) und es stellt der Anwendersoftware eine abstrakte Schnittstelle (die Systemaufrufschnittstelle) zu deren Nutzung zur Verfügung.

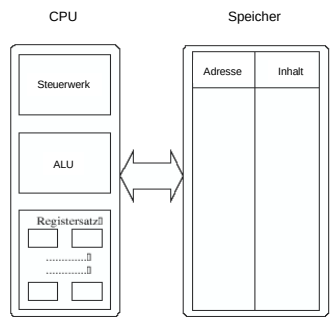
Dadurch vereinfacht es die Nutzung der Ressourcen und schützt vor Fehlbedienungen.

- Betriebssysteme, die es mit diesem Schutz nicht so genau nehmen, führen zu häufigen **Systemabstürzen** (system crash).



Kern aller heutigen Computer

Grundsätzlicher Aufbau verschiedener Rechnersysteme ist ähnlich:



Wahlfreier Zugriff

Von Neumann Architektur

Speicher

Kleinste Speichereinheit hat 2 Zustände

- 1 Bit
- Zustände werden i.A. mit 0 und 1 bezeichnet

Mit 2 Speichereinheiten 2^2=4 Zustände darstellbar

Bit 1	Bit 0	
0	0	Zustand 0
0	1	Zustand 1
1	0	Zustand 2
1	1	Zustand 3

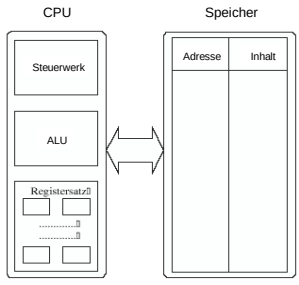
Mit 8 Bit 2^8=256 Zustände darstellbar

- 8 Bit = 1 Byte
- Heutzutage sind Bytes die kleinsten *adressierbaren* Speichereinheiten
  - Kleinere Einheiten müssen aus einem Byte extrahiert werden

z.B. 0010 0101

Speicher & Adressierung

Von Neumann Architektur



Adresse	Inhalt (byte)
1.	0101 0010
2.	1100 1100
3.	1001 1000
4.	0000 0100
5.	1111 1001
6.	0010 1011
7.	.....
..	

Wieviel Speicheradressen hat ein Computer?

Maximal soviel der Datenbus codieren kann.

- PC's mit 32 Bit Datenbus
  $2^{32} = 4 * 2^{30} = 4 * 2^{10} * 2^{10} * 2^{10}$ = 4 Giga Zustände (Adressen)
- bei 64 Bit Datenbus $2^{64} = 2^{34}$ Giga $\approx 10^{11}$ Giga Zustände (Adressen)

Daten und Befehle

Daten — Menge addressierbarer Speicherzellen

Daten sind binär gespeichert (z.B. 17 = 10001)
Binärspeicherung ist universel (Zahlen, Texte,Filme, Ton …)
1 Byte := 8 Bit
1 Wort := 2 Byte (z.T. auch 4 Byte)

Befehle — Operationen mit den Speicherzellen

Maschinensprache		Hochsprache
ACC ← x	// Lade Zelle x	z = x + y
ACC ← ACC + y	// Addiere y	
z ← ACC	// Speichere Ergebnis in Zelle z	

Variablen und der Typ von Variablen

Menschen benennen Dinge gerne mit Namen statt mit numerischen Adressen, so kennt jede Programmiersprache das Konzept einer **Variablen** als abstraktes Analogon zu einer Speicherstelle.

Eine Variable hat einen symbolischen **Namen**,

- hinter dem eine **Adresse** verborgen ist,
- und der **Wert** (value) der Variable ist der Wert des dort gespeicherten Bitmusters.
  - Um diesen erschließen zu können, hat die Variable einen **Typ** (type), der bei ihrer Vereinbarung angegeben werden muss.

Binärcodierung elementarer Datentypen

Unterscheide

- Zahl-Wert
- Zahl-Bezeichner

Zu ein- und demselben Zahl-Wert kann es verschiedene Bezeichner geben, z. B.

- Fünf, 5, V, 101

Da es unendlich viele Zahl-Werte gibt, ist es sinnvoll, sich eine Systematik zur Erzeugung von eindeutigen Bezeichnern zu schaffen

- Die auch das Rechnen mit Zahlen unterstützt
  - Verwendung von römischen Zahlen bietet keine gute Unterstützung

Binärcodierung elementarer Datentypen

Ein Zahlsystem (number system) besteht aus

- endlich vielen Ziffern (digits) und
- einer Vorschrift,
  - wie Zeichenreihen, die aus diesen Ziffern gebildet wurden, als Zahl-Werte zu interpretieren sind

Arabische Zahlsysteme zur Basis β

- Natürliche Zahl z wird geschrieben als Polynom

$$z = \sum_{i=0}^{n-1} z_i \beta^i$$

  - Dabei $0 \leq z_i < \beta$

Datentypen & Variable

Adresse	Inhalt (byte)
1. 0000 0000 0000 0000 … 0000 0000 0000 0000	0101 0010
2. 0000 0000 0000 0000 … 0000 0000 0000 00001	1100 1100
3. 0000 0000 0000 0000 … 0000 0000 0000 00010	1001 1000
4. 0000 0000 0000 0000 … 0000 0000 0000 00011	0000 0100
5. 0000 0000 0000 0000 … 0000 0000 0000 00100	1111 1001
6. 0000 0000 0000 0000 … 0000 0000 0000 00101	0010 1011
7. 0000 0000 0000 0000 … 0000 0000 0000 00111	.....
..	

Wie ist der Inhalt des Speichers zu interpretieren?

Dem Bitmuster ist seine Bedeutung (Text, Zahlen oder Musik) nicht fest zugeordnet.

Je nach Variablendefinition müssen unterschiedlich viele Bytes interpretiert werden und gleiche Bitmuster können unterschiedliche Bedeutungen haben!

Variablen haben immer einen Datentypen

Elementare Datentypen in Java

byte	8 Bit Zahl $-2^7 \ldots 2^7-1$ (-128, …. , 127)
short	16 Bit-Zahl $-2^{15} \ldots 2^{15}-1$ (-32768, …. , 32767)
int	32 Bit-Zahl $-2^{31} \ldots 2^{31}-1$ (-2 147 483 648, …. , 2 147 483 647)
long	64 Bit-Zahl $-2^{63} \ldots 2^{63}-1$
float	32 Bit IEEE-754-1985 Gleitkommazahl
double	64 Bit IEEE-754-1985 Gleitkommazahl
char	16 Bit Unicode
boolean	Wahrheitswert, *false* oder *true*

Gleitkommazahlen engl. „Floating-Point“

Darstellung einer Floating-Point-Zahl: $z = (-1)^v \cdot \text{Mantisse} \cdot 2^{\text{Exponent}}$

Floating-Point-Zahlen nach IEEE 754-1985

- 32 Bit `float`
- 64 Bit `double`

float	v	30 - Exponent - 23	22 - Mantisse - 0
	1 Bit	8 Bit	23 Bit
double	v	62 - Exponent - 52	51 - Mantisse - 0
	1 Bit	11 Bit	52 Bit

Algorithmus

Schrittweises, präzises Verfahren zur Lösung eines Problems

Problem: Summiere die Zahlen von 1 bis *max*. $sum = \sum_{i=1}^{i=\max} i$

Name Parameter

SummiereZahlenVon1bis*max* (↓ *max*, ↑*sum*)

1. *sum* ← 0
2. *zahl* ← 1
3. Wiederhole, solange *zahl* ≤ *max* — Folge
   - 3.1 *sum* ← *sum* + *zahl*
   - 3.2 *zahl* ← *zahl* + 1

Programm = Beschreibung eines Algorithmus in einer Programmiersprache

Variablen

Sind benannte Behälter für Werte

Können ihren Wert ändern

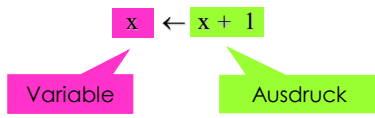
x ← x + 1

Haben einen Datentyp = Menge erlaubter Werte

Variablentyp	Werte	
Zahl	17 54 ...	- in eine Zahlenvariable passen nur Zahlen
Zeichen	'a' 'x' ...	- in eine Zeichenvariable passen nur Zeichen

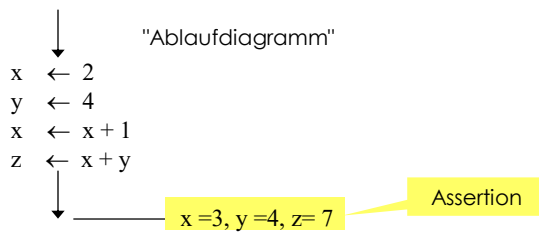
Anweisungen

Wertzuweisung



1. Werte Ausdruck aus
2. Weise seinen Wert der Variablen zu:

Anweisungsfolge (auch Sequenz)

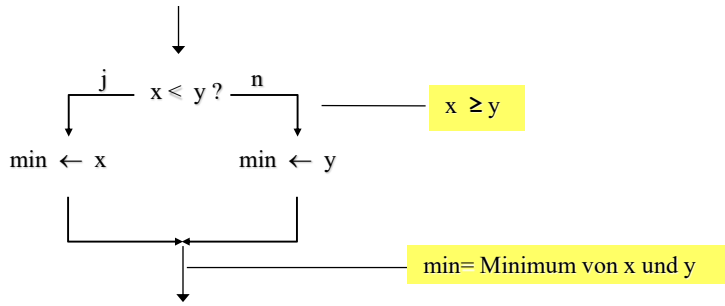


Assertion (Zusicherung)
Aussage über den Zustand des Algorithmus an einer bestimmten Stelle

Anweisungen

Auswahl (auch Verzweigung, Abfrage, Selektion)

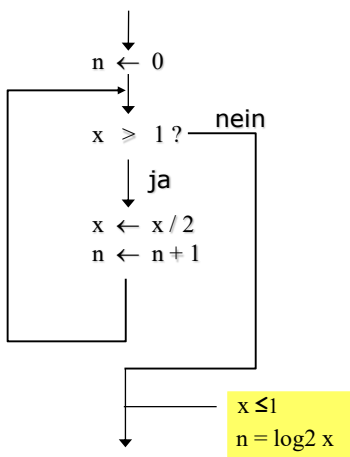
Beispiel: Suche das Minimum der zwei Zahlen x und y.

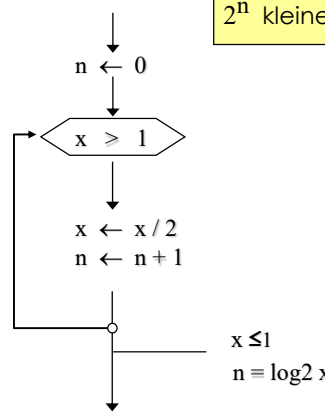


Anweisungen

Wiederholung (auch Schleife, Iteration)

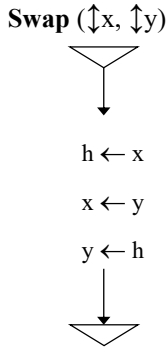
Beispiel: Suche die grösste ganze Zahl n mit 2^n kleiner oder gleich x .





Alternative Darstellung

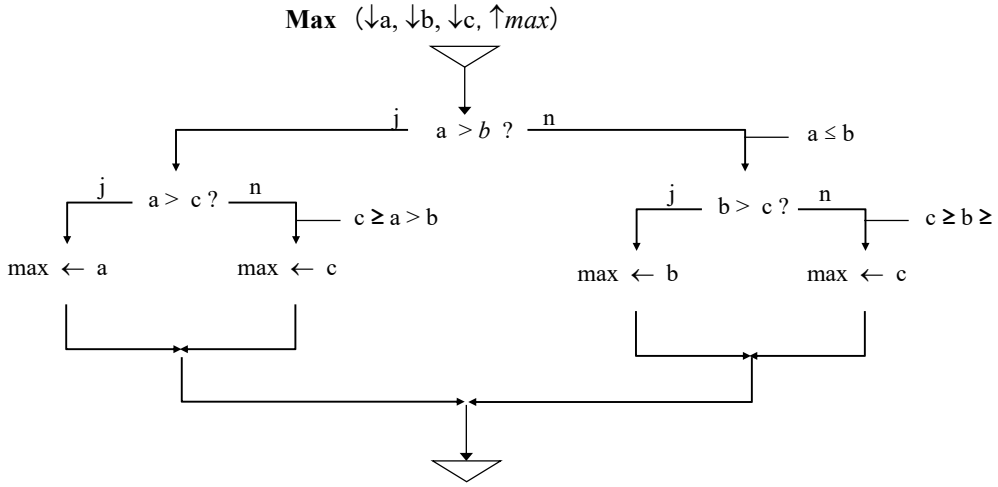
Beispiel: Vertausche zwei Variableninhalte



Schreibtischtest

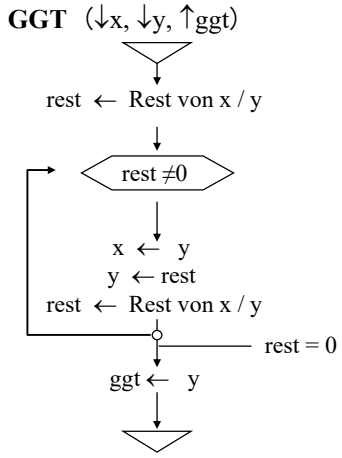
x	y	h
~~3~~	~~2~~	3
2	3	

Beispiel: Bestimme Maximum dreier Zahlen



Beispiel: Euklidscher Algorithmus

Berechnet den größten gemeinsamen Teiler zweier Zahlen x und y



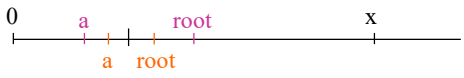
Schreibtischtest

x	y	rest
~~28~~	~~20~~	8
~~20~~	~~8~~	4
8	4	0

Warum funktioniert dieser Algorithmus?

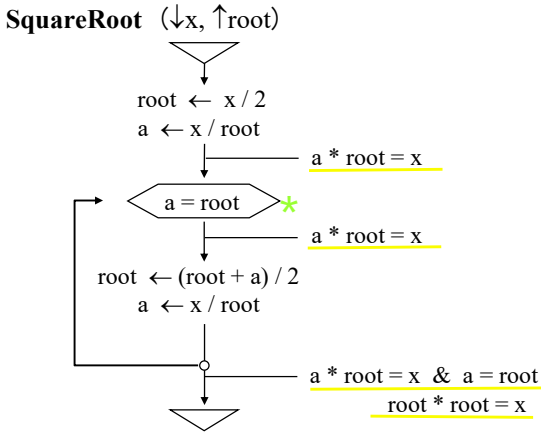
(*ggt* teilt x) & (*ggt* teilt y)
ggt teilt (x - y)
ggt teilt (x - q* y)
ggt teilt rest
ggt(x, y) = *ggt*(y, rest)

Beispiel: Berechne Quadratwurzel von x



1. Näherung:
root ← x / 2
a ← x / root

2. Näherung:
root ← (root + a) / 2
a ← x / root



Schreibtischtest

x	root	a
10	5	2
	3,5	2,85471
	3.17857	3,14607
	3.16232	3,16223
	3.16228	3,16228

Kommazahlen sind meist nicht exakt gleich, daher besser

| a- root | < 0.0000001