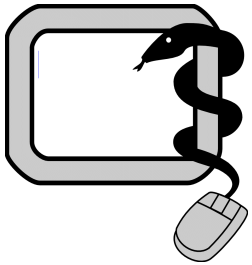
# Themen der Medizinischen Informatik: Seminar 2008/2009

## Grundbegriffe der Informatik

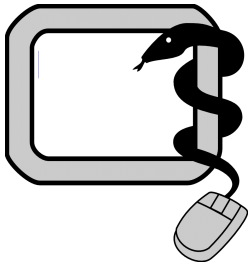
Prof. Dr. med.
Stefan Schulz

Institut für Medizinische Biometrie
und Medizinische Informatik



# Ziele der Lehrveranstaltung

- Erarbeitung der folgenden Grundbegriffe der Informatik:
  - Algorithmus
  - Programmierung
  - Datenstruktur
  - Datenbank
- Praktische Übungen

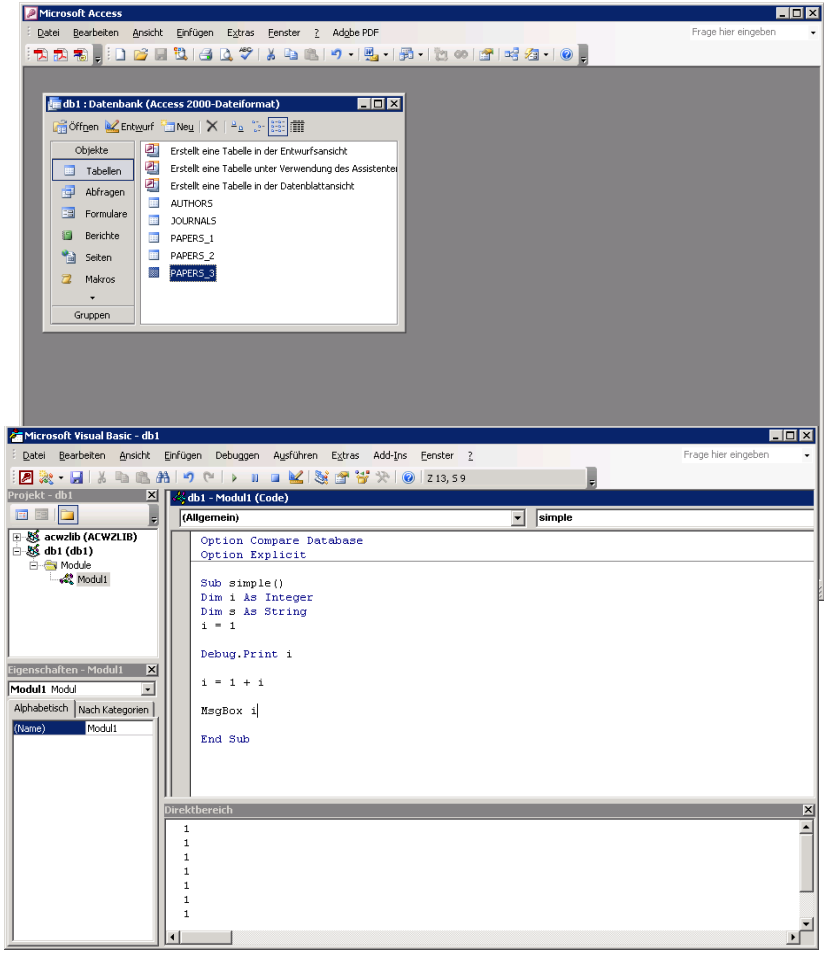


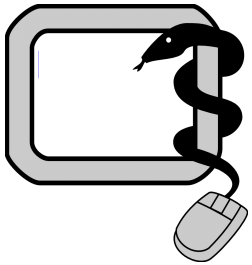
# Struktur

| | | |
|---|---|---|
| Grundkonzepte | Implementation | Entwicklungsumgebung |

# Werkzeuge

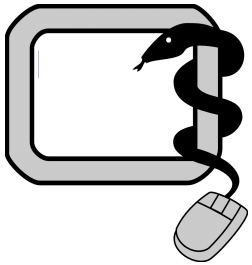
- Microsoft Access
  - Datenbanksystem erlaubt die Erstellung von Tabellen, Abfragen
  - Programmierschnittstelle erlaubt die Erstellung von Programmcode mit oder Zugriff auf Datenbankinhalte





# Algorithmus

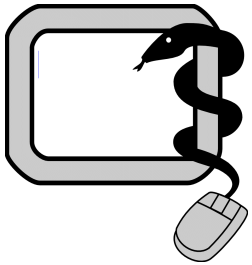
- genau definierte Handlungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen in endlich vielen Schritten.
- in einem endlichen Text eindeutig beschreibbar sein
- Jeder Schritt des Verfahrens muss tatsächlich ausführbar sein
- darf zu jedem Zeitpunkt nur endlich viel Speicherplatz benötigen
- darf nur endlich viele Schritte benötigen
- Der Algorithmus muss bei denselben Voraussetzungen das gleiche Ergebnis liefern (Determiniertheit).
- Die nächste anzuwendende Regel im Verfahren ist zu jedem Zeitpunkt eindeutig definiert (Determinismus).



# Algorithmen im Alltag

| | | |
|---|---|---|
| 500 | g | Kalbshackfleisch |
| 2 | St | Zwiebel |
| 1 | St | Ei |
| 1 | TL | Salz |
| 1/4 | TL | Pfeffer |
| 1 | EL | getrocknete Petersilie |
| 4 | TL | Senf |
| 1 | EL | Paniermehl |
| 2 | l | Gemüsebouillon |
| 1/2 | TL | Salz |
| 1 | EL | Butter |
| 1 | EL | Mehl |
| 1 | St | Eigelb |
| 1/2 | Becher | Schmand |
| 1 | gl | Kapern (kleines Glas) |

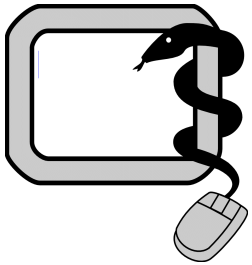
Das Kalbshack (ersatzweise auch gemischtes Hackfleisch) in eine Schüssel geben, die Zwiebeln fein würfeln und zugeben. Das Ei, Petersilie, Salz, Pfeffer und Senf zugeben und zu einer glatten Masse verarbeiten. Nun das Paniermehl daruntermengen. Die Bouillon aufkochen, den 1/2 Tel. Salz zugeben, einen kleinen Probekloß formen und in der Bouillon garen.
Eventuell noch etwas nachwürzen oder noch Paniermehl zugeben. Die Hackmasse in ca. 5-6 cm große Klöße formen und in der leicht sprudelnden Bouillon garen.
Wenn die Klöße oben schwimmen noch ca. 5 Min. ziehen lassen, abschöpfen und in ein Sieb geben.
Die Butter leicht braun werden lassen das Mehl unterrühren und von der Kochstelle nehmen.
Ca. 3/4 l. von der Fleischbrühe unterrühren, schnell mit dem Schneebesen eingeben. Wenn die Soße eine cremige Konsistenz erreicht hat ist genug Brühe darin.
Nun den Schmand zugeben und das Eigelb (mit einem Eßl. Milch verrührt) schnell einrühren, die Kapern zugeben, nochmals abschmecken und fertig.



# Steuerstrukturen (I)

- Bedingte Anweisung (if… then…else)

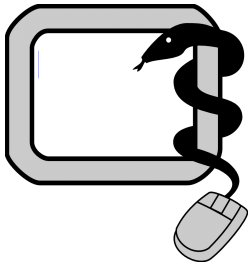
*Wenn die Zwiebeln klein*
    *nimm drei Stück*
*Ansonsten*
    *nimm zwei Stück*
*Ende*



# Steuerstrukturen (II)

- Zählschleife

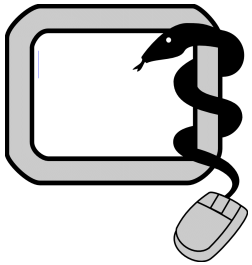
*Öffne das Senfglas*
*Für n von 1 bis 5*
*entnimm den n-ten Löffel*
*Ende*



# Steuerstrukturen (II)

- Kopfgesteuerte (abweisende) Schleife
  *Solange Soße fade*
  *eine Prise Salz zugeben*
  *Ende*

- Fußgesteuerte (durchlaufende) Schleife
  *Begin*
  *eine Prise Salz zugeben*
  *Bis Soße schmackhaft*

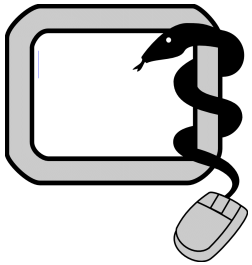


# Steuerstrukturen (III)

- Modularisierung:

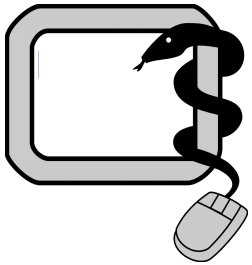
*Die Bouillon aufkochen*

*Die Bouillon in einen Kochtopf geben. Kochtopf auf Herdplatte stellen. Herdplatte auf Stufe 3 stellen. Sobald Dampfblasen aufsteigen, Herdplatte abschalten*



# Aufgabe

- Finden Sie Algorithmen aus dem Alltag und identifizieren Sie
  - Bedingte Anweisungen
  - Schleifen
  - geschachtelte Algorithmen



# Mathematische Algorithmen

- Berechne die Summe der Zahlen 1 bis n

*setze Summe 0*

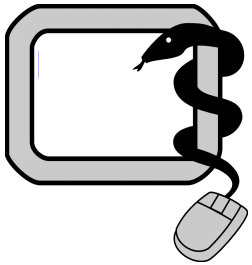
*setze Zähler 1*

*solange Zähler ≤ 1*

*ersetze Summe durch summe + Zähler*

*erhöhe Zähler um 1*

*gib aus: „Die Summe ist: “ und Summe*

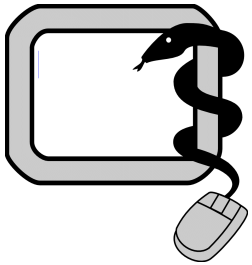


# Mathematische Algorithmen

- Sieb des Eratosthenes: Bestimme alle Primzahlen <= n

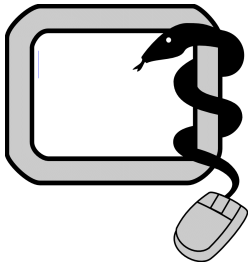
| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
| 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 |
| 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 |
| 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 |
| 61 | 62 | 63 | 64 | 65 | 66 | 67 | 68 | 69 | 70 |
| 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 |
| 81 | 82 | 83 | 84 | 85 | 86 | 87 | 88 | 89 | 90 |
| 91 | 92 | 93 | 94 | 95 | 96 | 97 | 98 | 99 | 100 |
| 101 | 102 | 103 | 104 | 105 | 106 | 107 | 108 | 109 | 110 |
| 111 | 112 | 113 | 114 | 115 | 116 | 117 | 118 | 119 | 120 |

**Primzahlen:**



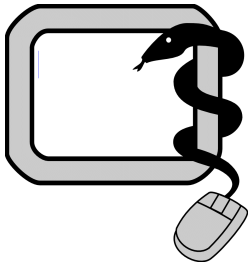
# Mathematische Algorithmen

- Sieb des Eratosthenes: Bestimme alle Primzahlen <= n
  - Input: Liste aller natürlichen Zahlen [2,…,n]
  - Output: Liste aller Primzahlen <= n
- Prinzip:
  - beginne mit der kleinsten Zahl: markiere alle Vielfache
  - gehe zur nächsten unmarkierten Zahl: markiere alle Vielfache
  - Abbruch bei $\sqrt{n}$



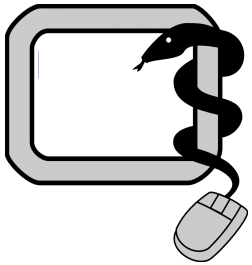
# Mathematische Algorithmen

- Beobachtung:
  - Algorithmus verarbeitet Daten
  - Daten sind Werte (z.B. „1“)
    oder Variablen (z.B. „Zähler“): „Behälter“ für Werte
  - Algorithmus



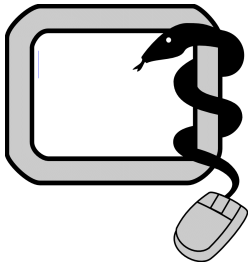
# Programm

- „Programm = Daten + Befehle“
- Beschreibung eines Ablaufes, der auf einer Rechenanlage durchgeführt werden kann (= Algorithmus, *statisch*).
- Programme sind (meist) in Dateien gespeichert.
- Programme entstehen durch Konstruktion.

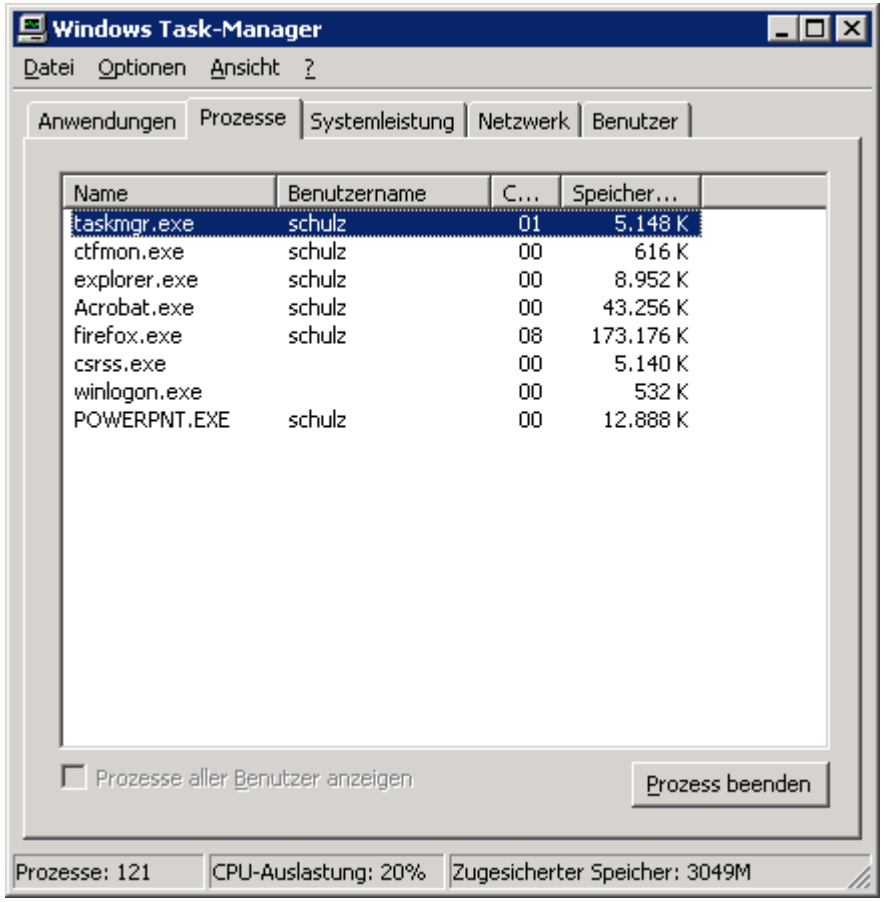


# Prozess

- Konkreter Ablauf eines Programms (*dynamisch*).
- Prozesse benötigen Betriebsmittel (Rechenzeit, Speicher, Dateizugriffe).
- Prozesse werden durch das Betriebssystem verwaltet.
- Prozesse entstehen durch Aufrufe von Programmen.

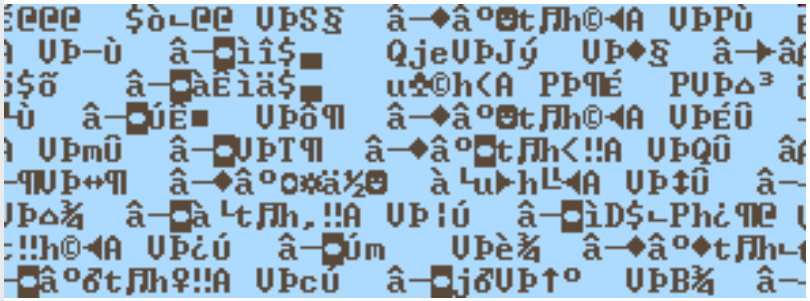


# Prozesse

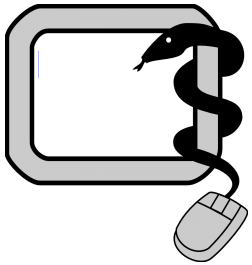

Windows Task-Manager
Datei Optionen Ansicht ?
Anwendungen Prozesse Systemleistung Netzwerk Benutzer
Name Benutzername C... Speicher...
taskmgr.exe schulz 01 5.148 K
ctfmon.exe schulz 00 616 K
explorer.exe schulz 00 8.952 K
Acrobat.exe schulz 00 43.256 K
firefox.exe schulz 08 173.176 K
csrss.exe 00 5.140 K
winlogon.exe 00 532 K
POWERPNT.EXE schulz 00 12.888 K
Prozesse aller Benutzer anzeigen
Prozess beenden
Prozesse: 121
CPU-Auslastung: 20%
Zugesicherter Speicher: 3049M

# Übersetzung von Programmen

```
class SummeBis {
public static void main (String[] arg) {
int n = Integer.parseInt(arg[0]);
int sum = 0; int i = 1;
while (i <= n) {sum = sum + i;i = i + 1;}
System.out.println („Die Summe ist: “ + sum);}}
```

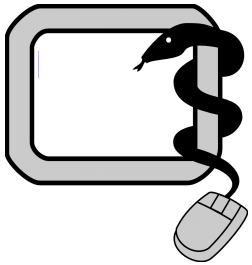


- Programme werden in höheren Programmiersprachen formuliert: Pascal, C, Java, Python, VB (Quellcode, Sourcecode)
- Die Hardware kann nur Programme in Maschinensprache ausführen.
- Abbildung höherer Programmiersprachen in Maschinensprachen nötig.



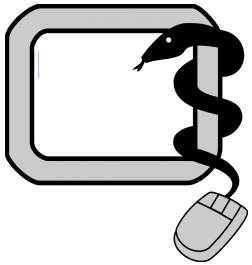
# Übersetzung von Programmen

- Zwei Vorgehensweisen
  - *Übersetzung:* Programme werden vor dem Ablauf übersetzt (kompiliert).
- – *Interpretation:* Programme werden während des Ablaufs interpretiert.
- Zwischenstufen:
  - Kompilieren zur Laufzeit
  - Übersetzung in plattformunabhängigen Bytecode (z.B.Java), dann just in time in Maschinencode



# Notationen

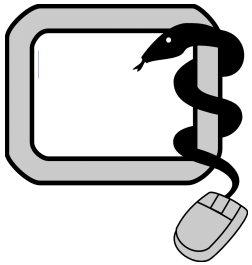
- Informell
  - Natürliche Sprache
  - Pseudocode (halbformell)
- Graphische Darstellungen
  - Programmablaufplan (Flussdiagramm, DIN 66001)
  - Struktogramm (Nassi-Shneiderman-Diagramm, DIN 66261)
- Formale Sprachen (Programmiersprachen)
  - Programm als formaler Text



# Beispiel: Summe von 1 bis n

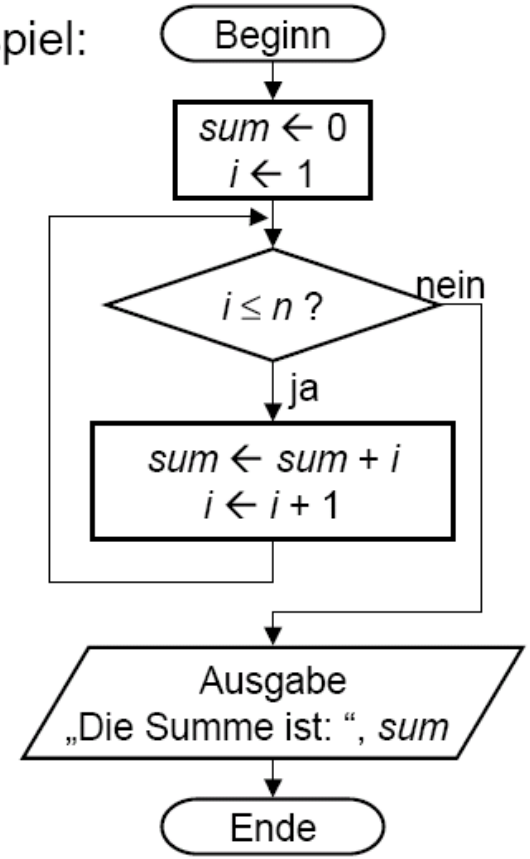
- Natürliche Sprache:
  - Initialisiere eine Variable *summe* mit 0. Durchlaufe die Zahlen von 1 bis *n* mit einer Variable *Zähler* und addiere *Zähler* jeweils zu *Summe*. Gib nach dem Durchlauf den Text “Die Summe ist: “ und den Wert von *Summe* aus.
- Pseudocode

```
setze summe := 0;
setze zähler := 1;
solange zähler ≤ n:
    setze summe := summe + zähler;
    erhöhe zähler um 1;
gib aus: “Die Summe ist: “ und summe;
```

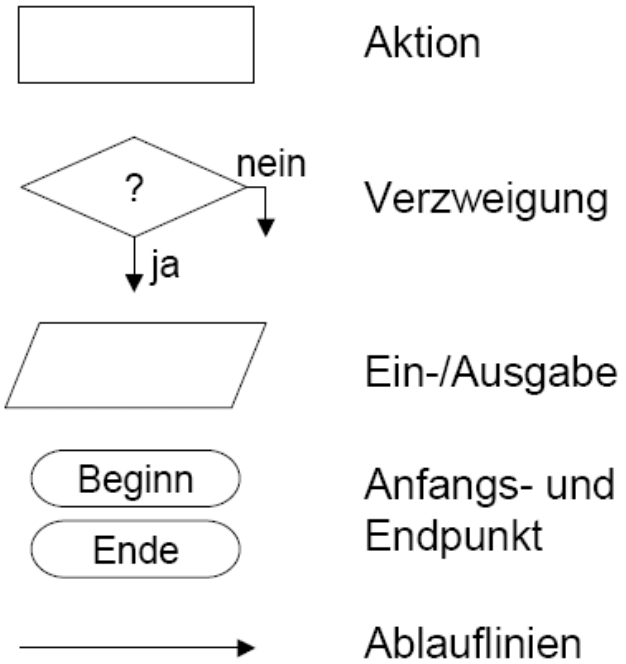


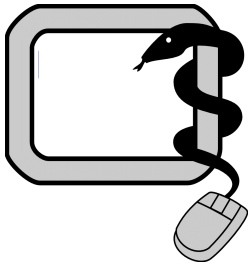
# Programmablaufplan

Beispiel:



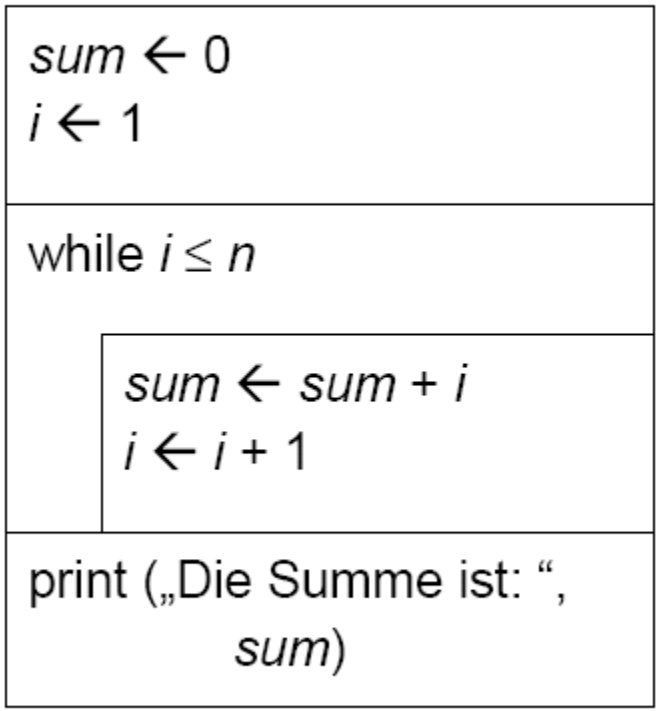

Symbole:



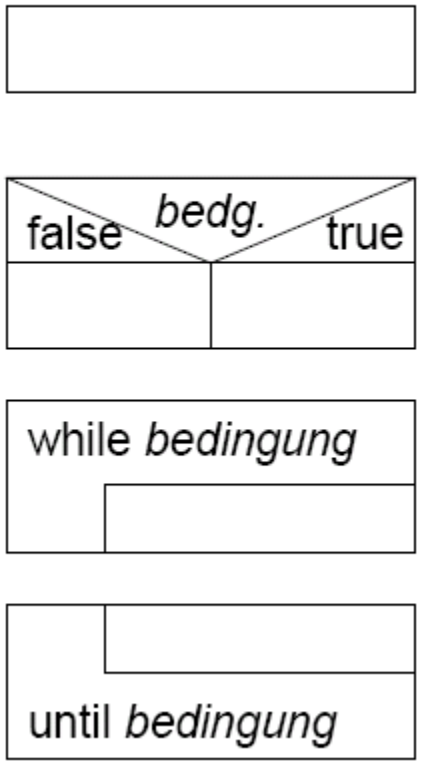


# Struktogramm

Beispiel:




Symbole:

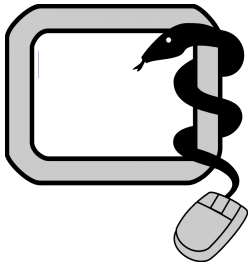



Strukturblock

Verzweigung

abweisende Schleife („*solange noch*“)

Durchlauf-Schleife („*solange bis*“)



# Programmiersprache

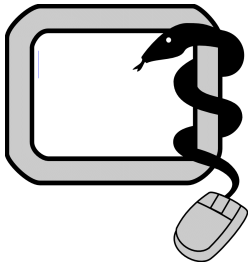
## JAVA

```
class SummeBis {
    public static void main (String[] arg) {
        int n = Integer.parseInt(arg[0]);
        int sum = 0;
        int i = 1;
        while (i <= n) {
            sum = sum + i;
            i = i + 1;
        }
        System.out.println („Die Summe ist: “ + sum);
    }
}
```

## Visual Basic

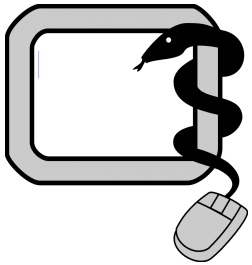
```
Sub SummeBis(n As Integer)
    Dim sum As Integer
    Dim i As Integer
    sum = 0
    i = 1
    Do While (i <= n)
        sum = sum + i
        i = i + 1
    Loop
    Debug.Print "Die Summe ist " & sum
End Sub
```

siehe Vergleich unter: http://dada.perl.it/shootout/



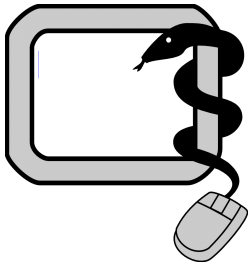
# Syntax und Semantik

- Natürliche Sprache
  - Syntax:
    „Der Hund bellt“ (syntaktisch korrekt)
    „Der Hund bellen“ (syntaktisch falsch)
  - Semantik:
    „Der Hund bellt“ (semantisch korrekt)
    „Der Mond bellt“ (semantisch falsch)



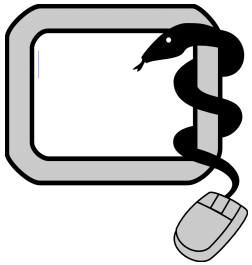
# Syntax und Semantik

- Syntax
  - Regeln, nach denen Programme aufgeschrieben werden dürfen.
  - Beispiel in Visual Basic: *Zuweisung ::= Variable = Ausdruck*
    - sum = 3 + 5
    - B = 2
    - sum = sum + 1
- Semantik
  - Bedeutung von korrekt gebildeten Programmen.
  - z.B.: *Weise den Wert des Ausdrucks an die Variable zu*
  - Nach „A = 3 + 5“ enthält A den Wert der Summe 3+5, also 8.



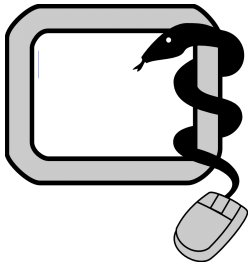
# Syntax und Semantik

- Finden Sie Beispiele für syntaktische und semantische Fehler in Programmen



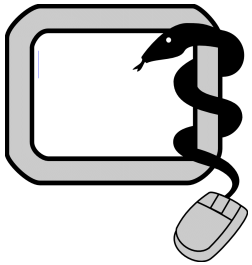
# Programmiersprache Visual Basic

- Einheitliche Programmierschnittstelle (Alt-F11) in MS-Office-Programmen aufrufbar
- Kode über Makrofunktion in Word, Excel, Powerpoint generierbar
- Für Anfänger geeignet
- Auch in größeren Projekten im Rahmen der .net – Plattform verwendbar
- Allerdings: von Microsoft kontrolliert !



# Aufbau einer einfachen VB-Routine (Demo)

- Header mit Parameterübergabe
- Variablendeklaration
- Programmablauf
- Eingabe: z.B. InputBox
- Ausgabe: z.B. MsgBox, Direktfenster
- Ein/Ausgabe



# Euklidischer Algorithmus

- Aufgabe: größter gemeinsamer Teiler zweier Zahlen
- Lösung von Euklid (um 300 v.Chr.)

- Euklid (in *a*, in *b*, out *ggt*):

```
rest ← a mod b
while rest ≠ 0
    a ← b
    b ← rest
    rest ← a mod b
ggt ← b
```

Nachweis der Korrektheit

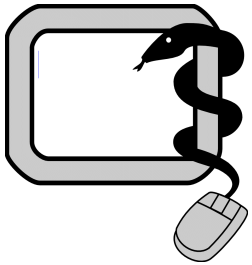
(ggt teilt *a*) ∧ (ggt teilt *b*)

➔ ggt teilt (*a* – *b*)

➔ ggt teilt (*a* – *k***b*)

➔ ggt teilt *rest*

➔ GGT (*a*,*b*) = GGT (*b*,*rest*)



# Maximum dreier Zahlen