Was ist Informatik?

Was machen Informatiker?
"Computer science is no more about computers than astronomy is about telescopes."

<table>
<thead>
<tr>
<th>Window</th>
<th>Central Concern</th>
<th>Principal Stories</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>What can be computed and how; limits of computing.</td>
<td>Algorithm, data structure, automata, languages, Turing machine, universal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>computer, Turing complexity, Chaitin complexity, self reference, approximations,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>heuristics, non-computability, translations, compilations, physical realizations</td>
</tr>
<tr>
<td>Communication</td>
<td>Sending messages from one point to another.</td>
<td>Data transmission, Shannon entropy, encoding to medium, channel capacity,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>noise suppression, error correcting codes, end-to-end-error correction,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Huffman and Reed-Solomon codes, file compression, cryptography, packet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>networking</td>
</tr>
<tr>
<td>Coordination</td>
<td>Multiple entities cooperating toward a single result.</td>
<td>Human-to-human (action loops, workflows as supported by communicating computers),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>human-computer (interface, input, output, response time, data visualization),</td>
</tr>
<tr>
<td></td>
<td></td>
<td>computer-computer (concurrency control, races, synchronization, deadlock,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>serializability, atomic actions)</td>
</tr>
<tr>
<td>Automation</td>
<td>Performing cognitive tasks by computer.</td>
<td>Simulation and machine performance of cognitive tasks, philosophical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>distinctions about automation, expertise and expert systems, enhancement of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>intelligence, Turing tests, machine learning and recognition, bionics</td>
</tr>
<tr>
<td>Recollection</td>
<td>Storing and retrieving information.</td>
<td>Hierarchies of storage, locality of reference, caching, address space and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>mapping, bindings, naming, sharing, thrashing, retrieval by name, retrieval by</td>
</tr>
<tr>
<td></td>
<td></td>
<td>content</td>
</tr>
<tr>
<td>Area</td>
<td>Problem</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
</tbody>
</table>
| Computation| • Unbounded error accumulation on finite machines
               • Non-computability of some important problems
               • Intractability of thousands of common problems
               • Optimal algorithms for some common problems
               • Production quality compilers                                      |
| Communication| • Lossless file compression                                              |
               • Lossy but high-fidelity audio and video compression
               • Error correction codes for high, bursty noise channels
               • Secure cryptographic key exchange in open networks               |
| Interaction | • Arbitration problem                                                  |
               • Timing-dependent (race-conditioned) bug problem
               • Deadlock problem                                                  |
               • Fast algorithms for predicting throughput and response time
               • Internet protocols                                                |
               • Cryptographic authentication protocols                           |
| Recollection| • Locality                                                              |
               • Thrashing                                                           |
               • Search                                                             |
               • Two-level mapping for access to shared objects                   |
| Automation  | • Simulations of focused cognitive tasks                               |
               • Limits on expert systems                                          |
               • Reverse Turing tests                                             |
| Design      | • Objects and information hiding                                       |
               • Levels                                                              |
               • Throughput and response time prediction networks of servers      |

---

<table>
<thead>
<tr>
<th>Principle</th>
<th>Summary</th>
<th>Computing Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compression (Communication)</td>
<td>Representations of data and algorithms can be significantly compressed and the most valuable information recovered later.</td>
<td>Compression of voice (MP3, MP4, ACC), images (JPEG, GIF), files (Zip). Fourier transform.</td>
</tr>
<tr>
<td>Choosing (Coordination)</td>
<td>An uncertainty principle: it is not possible to make an unambiguous choice of one of several alternatives within a fixed deadline.</td>
<td>Hardware that never crashes while responding to interrupts. Mutual exclusion. Deadlocks.</td>
</tr>
<tr>
<td>Bottlenecks (Evaluation)</td>
<td>Forced flow laws: in any network, the throughput at any node is the product of the network throughput and the visits per task to the node.</td>
<td>Saturation and bottlenecks in communication networks.</td>
</tr>
</tbody>
</table>
### Principles of Computing

<table>
<thead>
<tr>
<th>Computation</th>
<th>Communication</th>
<th>Coordination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluation</td>
<td>Design</td>
<td></td>
</tr>
</tbody>
</table>

**Computation**
- Intractability (Computation)

**Communication**
- Compression (Communication)

**Coordination**
- Choosing (Coordination)

**Recollection**
- Locality (Recollection)

**Automation**
- Search (Automation)

**Evaluation**
- Bottlenecks (Evaluation)

**Design**
- Hierarchical Aggregation (Design)

---

Principle

Computation

Wie werden Informationen repräsentiert?
Wie werden Berechnungen repräsentiert?

Was ist berechenbar?
Was ist nicht berechenbar?
Computer science was born in 1936, at the dawn of the electronic digital computer, when Alan Turing wrote about the capabilities of computing machines. [...] 

In 1936, Alan Turing published a famous paper, “On the computable numbers, with an application to the Entscheidungsproblem.” Turing defined computation, computing machines, and universal machines, and he nonchalantly showed that the halting problem for machines was not computable. From that conclusion he demonstrated that the century-old “Entscheidungsproblem” (German for decision problem) had no solution. That problem posited a complete and consistent universal logic system that would be able to tell if a proposition from any other logic system is true. It was on the mathematician David Hilbert’s 1928 list of challenge problems in mathematics. It dreamt of a method to tell “by inspection” whether a computation halted. Turing showed that if such a logic system existed, it would be able to answer the halting question. He concluded that the dream of a “by inspection” method to answer halting questions was impossible. He showed that the very steps a mathematician might use to apply a “by inspection” method were fundamentally computational. Therefore the only general method of approaching the halting question is to run the computations and see what happens. **Turing thus showed that computation is unavoidable.** This truly was the birth of computer science.

Peter J. Denning, Craig Martell. 
Practices of Computing

- Innovating
- Modeling and validation
- Engineering Systems
- Programming
Great Practices of Computing

Programming
Using programming languages to build software systems that meet specifications created in cooperation with the users of those systems. Computing professionals must be multilingual, facile with the numerous programming languages, each attuned to its own strategies for solving problems.

Engineering Systems
Designing and constructing systems of software and hardware components running on servers connected by networks. These practices include a design component concerned with organizing a system to produce valuable and tangible benefits for the users; and an engineering component concerned with the modules, abstractions, revisions, design decisions, and risks in the system; and an operations component concerned with configuration, management, and maintenance of the system. High levels of skill are needed for large programmed systems encompassing thousands of modules and millions of lines of code.
Great Practices of Computing

Modeling and validation
Building models of systems to make predictions about their behavior under various conditions; and designing experiments to validate algorithms and systems.

Innovating
Bringing about lasting changes in the ways groups and communities operate by exercising technical leadership. Innovators watch for and analyze opportunities, listen to customers, formulate offers customers see as valuable, and manage commitments to deliver the promised results. Innovators are history-makers who have strong historical sensibilities.
Programming

Bash

# Bash: "Hello, world!"
echo Hello, world!

BASIC

10 REM BASIC: "Hello, world!"
20 PRINT "Hello, world!"

Brainfuck


BS2000

/BEGIN-PROCEDURE LOGGING=N
/REMARK BS2000 (SDF): "Hello, world!"
/ASSIGN-SYSDTA TO-FILE=*SYSCMD
/WRITE-TEXT 'Hello, world!'
/SET-JOB-STEP
/ASSIGN-SYSDTA TO-FILE=*PRIMARY
/END-PROCEDURE
Knackpunkt – das umstrittene Wesen der Informatik – nur schon das Wesen des Programmierens ist umstritten.

Für manche ist Programmieren eine Art Kunst.

Für manche ist es eine wissenschaftliche Tätigkeit.

Für manche ist es ein Handwerk.

Für manche ist es eine Ingenieur-Tätigkeit.
Software Craftsmanship

The New Imperative
Computing!
Knackpunkt – das umstrittene Wesen der Informatik – nur schon das Wesen des Programmierens ist umstritten.

Was macht der Praktiker, wenn die Theoretiker sich nicht einig sind?

Er praktiziert ganz pragmatisch.
The Pragmatic Programmer

from journeyman to master
<table>
<thead>
<tr>
<th>Task</th>
<th>Hours per Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work</td>
<td>11</td>
</tr>
<tr>
<td>Eat</td>
<td>2</td>
</tr>
<tr>
<td>Commute</td>
<td>2</td>
</tr>
<tr>
<td>Watch TV</td>
<td>2</td>
</tr>
<tr>
<td>Sleep</td>
<td>7</td>
</tr>
</tbody>
</table>

So, how was your day?
function drawVisualization()
{
    // Create and populate the data table.
    var data = new google.visualization.DataTable();
    data.addColumn('string', 'Task');
    data.addColumn('number', 'Hours per Day');
    data.addRows([5]);
    data.setValue(0, 0, 'Work');
    data.setValue(0, 1, 11);
    data.setValue(1, 0, 'Eat');
    data.setValue(1, 1, 2);
    data.setValue(2, 0, 'Commute');
    data.setValue(2, 1, 2);
    data.setValue(3, 0, 'Watch TV');
    data.setValue(3, 1, 2);
    data.setValue(4, 0, 'Sleep');
    data.setValue(4, 1, 7);

    // Create and draw the visualization.
    new google.visualization.PieChart(document.getElementById('visualization')).
        draw(data, {title:'So, how was your day?'})
}
Übung am PC

Kuchendiagramme
Practice Engineering Systems
WANTED
BY U.S. MARSHALS

MITHIK, KEVIN DAVID

DESCRIPTION:
Sex: MALE
Race: WHITE
Place of Birth: VAN NUYS, CALIFORNIA
Date(s) of Birth: 08/06/63; 10/18/70
Height: 5'11"
Weight: 190
Eyes: BLUE
Hair: BROWN
Skin tone: LIGHT
Scars, Marks, Tattoos: KNEE KNOWN
Social Security Number (s): 550-39-5695
NCIC Fingerprint Classification: DCM20PM13IPM19PM09

ADDRESS AND LOCATE: KNOWN TO RESIDE IN THE SAN FERNANDO VALLEY AREA OF CALIFORNIA AND LAS VEGAS, NEVADA

Wanted for: Computer fraud.
Software Entwickler: Keine Einzelkämpfer
Software Entwickler: Keine Einzelkämpfer

Arbeit alleine

Arbeit im Team

Arbeit zu zweit
Practice Engineering Systems

Ein Beispiel
Practice

Engineering Systems

Information Hiding

Practice Engineering Systems
Konzept Schnittstellen
function initialize() {
  if (BrowserIsCompatible()) {
    var map = new GMap2(document.getElementById("map_canvas"));
    map.setCenter(new GLatLng(37.4419, -122.1419), 13);

    // Add 10 markers to the map at random locations
    var bounds = map.getBounds();
    var southWest = bounds.getSouthWest();
    var northEast = bounds.getNorthEast();
    var lngSpan = northEast.lng() - southWest.lng();
    var latSpan = northEast.lat() - southWest.lat();
    for (var i = 0; i < 10; i++) {
      var point = new GLatLng(southWest.lat() + latSpan * Math.random(),
                              southWest.lng() + lngSpan * Math.random());
      map.addOverlay(new GMarker(point));
  }
}
Practice
Engineering Systems

Übung am PC
Maps für die eigene Webseite
Principle
Recollection: Search
Principle

Recollection: Search

Einschränken der Suche
Suchmaschine

1. Crawling/Spidering
   Download und Speicherung aller Dokumente der Kollektion

2. Erstellen Index
   Erstellen einer effizienten Datenstruktur für die Suche

3. Benutzeranfrage
   Finden passender Dokumente auf eine Benutzeranfrage

4. Resultate
   Präsentation der Treffer in der richtigen Reihenfolge

J. Stuker, namics ag
### Der Index einer Suchmaschine

<table>
<thead>
<tr>
<th>Begriffe</th>
<th>Vorkommen</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>mars</td>
<td>chocolate.com/mars.html</td>
<td>geschichte.de/mars.htm</td>
</tr>
<tr>
<td></td>
<td>120</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td>7, 12, 51, ...</td>
<td>21, 33, ...</td>
</tr>
<tr>
<td>pluto</td>
<td>disney.com/comics.html</td>
<td>planets.org/list.htm</td>
</tr>
<tr>
<td></td>
<td>78</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>1, 8, ...</td>
<td>67, 73, ...</td>
</tr>
<tr>
<td>saturn</td>
<td>sega.com/consoles.html</td>
<td>cars.uk/dealers.html</td>
</tr>
<tr>
<td></td>
<td>99</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>1, 4, 9, ...</td>
<td>51, 126, ...</td>
</tr>
</tbody>
</table>

---

Werner Hartmann, Michael Näf, Peter Schäuble (2000). Informationsbeschaffung im Internet; Grundlegende Konzepte verstehen und umsetzen. Orell Füssli Verlag Zürich.
So arbeitet die Suchmaschine mit dem Index

Gesucht
Übersicht über die Planeten unseres Sonnensystems

Anfrage
mars

Resultat
Alle Webseiten zum Begriff „mars“ aus dem Index, kaum etwas zu Planeten

Werner Hartmann, Michael Näf, Peter Schäuble (2000). Informationsbeschaffung im Internet; Grundlegende Konzepte verstehen und umsetzen. Orell Füssli Verlag Zürich.
So arbeitet die Suchmaschine mit dem Index

**Anfrage:** *mars*

<table>
<thead>
<tr>
<th>Begriffe</th>
<th>Vorkommen</th>
</tr>
</thead>
<tbody>
<tr>
<td>mars</td>
<td>chocolate.com/mars.html, geschichte.de/mars.htm, planets.org/list.htm, ...</td>
</tr>
<tr>
<td></td>
<td>120, 36, 3, 7, 12, 51, 21, 33, 12, 33, 40, ...</td>
</tr>
<tr>
<td>pluto</td>
<td>disney.com/comics.html, planets.org/list.htm, ...</td>
</tr>
<tr>
<td></td>
<td>78, 15, 67, 73, 12, 33, 40, ...</td>
</tr>
<tr>
<td>saturn</td>
<td>sega.com/consoles.html, cars.uk/dealers.html, planets.org/list.htm, ...</td>
</tr>
<tr>
<td></td>
<td>99, 10, 8, 51, 126, 80, 85, ...</td>
</tr>
</tbody>
</table>
So arbeitet die Suchmaschine mit dem Index

**Gesucht**
Übersicht über die Planeten unseres Sonnensystems

**Anfrage**
mars pluto saturn

**Resultat**
Anfrage: *mars pluto saturn*

<table>
<thead>
<tr>
<th>Begriffe</th>
<th>Vorkommen</th>
</tr>
</thead>
<tbody>
<tr>
<td>mars</td>
<td>chocolate.com/mars.html 120 7, 12, 51, ...</td>
</tr>
<tr>
<td></td>
<td>geschichte.de/mars.htm 36 21, 33, ...</td>
</tr>
<tr>
<td></td>
<td>planets.org/list.htm 3 12, 33, 40, ...</td>
</tr>
<tr>
<td>pluto</td>
<td>disney.com/comics.html 78 1, 8, ...</td>
</tr>
<tr>
<td></td>
<td>planets.org/list.htm 15 67, 73, ...</td>
</tr>
<tr>
<td></td>
<td>... ...</td>
</tr>
<tr>
<td>saturn</td>
<td>sega.com/consoles.html 99 1, 4, 9, ...</td>
</tr>
<tr>
<td></td>
<td>cars.uk/dealers.html 10 51, 126, ...</td>
</tr>
<tr>
<td></td>
<td>planets.org/list.htm 8 80, 85, ...</td>
</tr>
</tbody>
</table>

Werner Hartmann, Michael Näf, Peter Schäuble (2000). Informationsbeschaffung im Internet; Grundlegende Konzepte verstehen und umsetzen. Orell Füssli Verlag Zürich.
### Der Index: wenig häufige Worte

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sex</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bier</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fussball</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Auto</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

....
....

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Iglu</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Gölä</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Börkin</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Neadron</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Seltene Suchbegriffe helfen der Suchmaschine

Werner Hartmann, Michael Näge, Peter Schäuble (2000). Informationsbeschaffung im Internet; Grundlegende Konzepte verstehen und umsetzen. Orell Füssli Verlag Zürich.
Sie wollen bessere Suchresultate?

Verwenden Sie viele und spezifische Suchbegriffe!

Werner Hartmann, Michael Näf, Peter Schäuble (2000). Informationsbeschaffung im Internet; Grundlegende Konzepte verstehen und umsetzen. Orell Füssli Verlag Zürich.
```javascript
/*
 * How to restrict a search to a Custom Search Engine
 * http://www.google.com/cse/
 */

google.load('search', '1');

function OnLoad()

// Create a search control
var searchControl = new google.search.SearchControl();

// Add in a WebSearch
var webSearch = new google.search.WebSearch();

// Restrict our search to pages from our CSE
webSearch.setSiteRestriction('017876662512465239146:omuauf_l5ve', 'lectures');

// Add the searcher to the SearchControl
searchControl.addSearcher(webSearch);
```
Recollection: Search Übung am PC
Eine eigene Google Suchmaschine