Impressum

Chefredakteur (V.i.S.d.P.):
StD Dr. Ludger Humbert
Redaktion: StD Christian F. Görlich
& Fachseminare Informatik Hamm und Arnsberg
Layout: Ludger Humbert (Vorlage von Torsten Bronger)

If Fase … auch im Netz (↘ humbert.in.hagen.de/iffase)
ISSN 1861-0498

If Fase

IF FASE: INFORMATIKFACHSEMINARE HAMM, ARNSBERG

Bildung
Fachseminare
Veranstaltungen
Informatiksysteme
Ausbildungsschulen

If Fase

Nr. 12 — HERAUSGEGEBEN VON DER IF FASE-REDAKTION — 1. Oktober 2006

Termine

Freitag, 13. Oktober 2006

Informatik in Dortmund – Get in Touch

ab 14:00
Universität Dortmund – Audimax

u. a. Informatik zwischen 9-13 Jahren
↘ www.informatikjahr-dortmund.de

Montag, 16. Oktober 2006

Informatikfachleitertreffen NW

10:00 – 16:00
Soest – Landesinstitut

2. Informatikfachleitertreffen NW – Schwerpunkt **Informatik in der Sekundarstufe I aus fachdidaktischer und seminardidaktischer Sicht**

Freitag, 3. November 2006

22. FIfF-Jahrestagung

Bremen – Haus der Wissenschaft / ZIMT

Auftaktveranstaltung
↘ fiff.informatik.uni-bremen.de/2006/index.html

KurzNotiert

(von Dr. Ludger Humbert)

Abkürzungen – Herrschaftswissen

Gerade Schüler zeigen sich vermeintlich kompetent, wenn sie mit Abkürzungen nur so um sich werfen. Dabei ist ihnen häufig nicht klar, welche Bedeutung eine konkrete Abkürzung hat. Im Web finden sich verschiedene Quellen, die Abkürzungen erläutern. ↘ research.silmaril.ie/acronyms stellt eine Quelle dar, die eine kategoriegestützte Auswahl von Erläuterungen liefert und eine Schnittstelle anbietet, über die unbekannte Abkürzungen hinzugefügt werden können.

* * *

LaTeX + Nokia 770 = Maemo-TeX

Um auf dem Nokia 770 LaTeX einsetzen zu können, wurde das in der If Fase Nr. 11 erwähnte Paket inzwischen durch ein [mehrfach] aktualisiertes Debianpaket ersetzt, das mit dem »Application Installer« installiert werden kann ↘ www.ham.nw.schule.de/pub/bscw.cgi/365355 Aktuelle Fassung: maemotex_0.1.0-6_armel.deb vom 30. September 2006 – Es basiert auf den aktuellen Beta-Versionen für pdftex 1.40.0-beta-20060928 und metapost Version 0.932 vom 28. August 2006.

* * *

LaTeX nach HTML + Nokia 770 = MaemoTtH

Die Konvertierung von LaTeX nach HTML kann mit verschiedenen Programmen erfolgen. TtH stellt eine sehr effiziente Möglichkeit zur Verfügung, die sehr einfach auf andere Systeme protiert werden kann. Daher wurde das Debianpaket MaemoTtH erstellt. Es kann ebenfalls mit dem »Application Installer« installiert werden: ↘ www.ham.nw.schule.de/pub/bscw.cgi/368584

LaTeX – Teil 12: Quelltexte von Programmen

In einer Reihe von Artikeln in der If Fase werden nützliche Elemente von LaTeX vorgestellt, die erprobt sind und bei der Arbeit der Informatiklehrerin eingesetzt werden.

(von Dr. Ludger Humbert) In den bisher vorgelegten zwölf Teilen der Artikelserie – Ausgaben 0 … 11: ↘ humbert.in.hagen.de/iffase/Archiv – finden Sie Hinweise und Anmerkungen zu den Themen: Installation, grundlegende Arbeitsweisen, Quellen zu Dokumentationen, Arbeit mit KOMA-Script, PSTricks, Erstellung von Arbeitsblättern, Struktogrammen, Automatengrafen, Elemente von UML, Barcodes, Formularerstellung, Zitieren, Abbildungen, ER-Diagramme. Beginnend mit der Ausgabe 9 (Zitieren – normgerecht) wird das Thema von Fragen bestimmt, die von den Referendarinnen gestellt werden. Damit soll der konkrete Ausbildungsnutzen nachhaltig verbessert werden.

Code rules

Bestandteil der informatischen Modellierung ist die Umsetzung in eine Programmiersprache oder zumindest programmiersprachliche Konstrukte. Soll Programmcode Bestandteil von Texten sein, so ist zu bedenken, welcher konkrete Anwendungsfall vorliegt. Um die Entscheidungsindung zu unterstützen, werden zwei verschiedene Anwendungsfälle dargestellt, die im Zusammenhang mit LaTeX sehr gut unterstützt werden. Dabei wird der triviale Ansatz, Quellcodestücke mit einem entsprechenden Font zu setzen, hier nicht betrachtet. Gründe für diese Entscheidung werden durch das Studium des Artikels offensichtlich.

LaTeX – Ansätze zur Unterstützung

Literate Programming

Die Anforderung zur Unterstützung der Quellcode**entwicklung** wurde vom TeX-Meister selbst (== Don Knuth) formuliert und mit **Literate Programming** beantwortet:

The main idea is to regard a program as a communication to human beings rather than as a set of instructions to a computer
Quelle: ↘ sunburn.stanford.edu/~knuth/cweb.html

↘ literateprogramming.com stellt eine Sammlung von Elementen bereit, die diesen Ansatz zur schrittweisen Verfeinerung einer Problemlösung dokumentierten (inkl. der Originalarbeit von Donald Knuth aus dem Jahr 1984). Ich habe den Eindruck, dass dieser Ansatz unterbewertet wurde und habe selbst einige Erfahrungen in der entwicklungsbegleitenden Dokumentation gesammelt. Zunächst ist hervorzuheben, dass es nur noch eine Quelle für den Quellcode gibt *die Dokumentation*.

Literate Programming – Werkzeuge

Eine Datei, die sowohl die Dokumentation, aber auch den Quellcode enthält, wird mit der Endung `nw` versehen (im Unterschied zur üblichen Endung von LaTeX `tex`. Dies ist sinnvoll, da aus dieser Datei in einem nächsten Schritt die verschiedene Dateien (unter anderem eine `tex`-Datei) extrahiert werden. Diese umfassen die zusammengebauten Quellcodeschnipsel und selbstverständlich auch die `tex`-Datei, die dann für das Setzen der Dokumentation verwendet wird. Die Dokumentation enthält – neben den Codeschnipseln – Hypertextfunktionen, die es zum Beispiel ermöglichen, von Quelltextelement zu Quelltextelement weiterzugehen. Auf diese Weise ist es möglich, den Quelltext auch in der Dokumentation zusammenhängend zu lesen. Außerdem werden Möglichkeiten bereitgestellt, den Quelltext mit weiteren Eigenschaften so zu versehen, dass automatisch Indizes angelegt werden können, etc.

Ursprünglich unterstützte Knuth mit seiner Lösung die Programmiersprache `Pascal`. Inzwischen wird eine grosse Anzahl von Programmiersprachen direkt unterstützt (unter anderem auch `Python`).

Arbeitsablauf

Mit einem Editor der Wahl wird – wie üblich – der LaTeX-Quellcode für die Dokumentation erstellt. Dabei gibt es eine Reihe von Möglichkeiten für die weitere Verarbeitung der Quelle. Diese sind zusammenfassend auf **einer** Seite dokumentiert ↘ www.eecs.harvard.edu/~nr/noweb/onepage.ps Quellcode wird mittels << und >> ausgezeichnet. Damit ist die prinzipielle Struktur einfach realisierbar. Auf der o. g. Seite finden sich einige Kommandos, die auf die Noweb-Datei angewendet werden können, um den Quellcode zu extrahieren, um die Dokumentation (inkl. Kreuzreferenzen, etc.) zu erstellen, u.v.a.m.

Bei der Eingabe des Quellcodes für eine `*.nw`-Datei ist darauf zu achten, dass (außerhalb des Programmtextes) »Möwchen« als Auszeichnung für die sogenannten Chunks (== Codeschnipsel) reserviert sind. Dies hat die Konsequenz, dass für die von mir zur Darstellung von Zitaten bevorzugten Möwchen jeweils mit dem Prefix »@« eingeleitet werden müssen, damit sie nicht als Quellcode interpretiert werden. Damit stellen @>> und @<< bei meinen Texten den Beginn und das Ende eines Zitats dar, wenn Noweb verwendet wird.

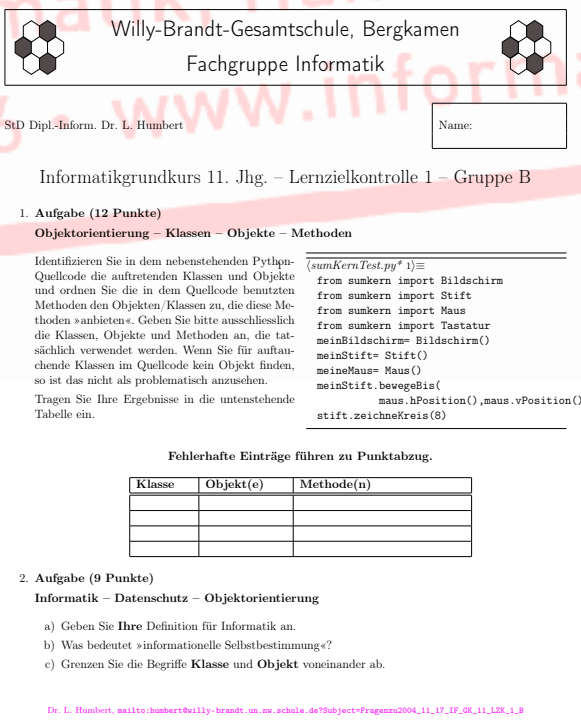
Unterstützung

LyX ↘ lyx.org unterstützt die Nutzung von Noweb, so dass die Dokumentation inklusiv Quellcode nicht unbedingt LaTeX-Kenntnisse erfordert, sondern mit der aus LyX bekannten WYSIWYM (What You See Is What You Mean) Sicht eingegeben werden kann. Bestandteil von LyX ist die Datei `noweb2lyx.lyx` die als Ausgangspunkt für eigene Testfälle herangezogen werden kann.

Mit **Leo** wurde ein Editor entwickelt, der von der Quellcodeseite ausgehend, die nötigen Funktionen über eine GUI bereitstellt. Benutzer berichten, dass sie mit diesem Ansatz eine völlig neue Sicht auf ihre Programmentwicklung gefunden haben. Ich habe dieses Werkzeug (das in Python geschrieben ist) bisher nicht eingesetzt. ↘ webpages.charter.net/edreamleo/front.html

Dokumentation inklusiv Quellcode

Der Ansatz, den Knuth mit Literate Programming verfolgt, besteht darin, den Entwicklungsprozess schrittweise so zu dokumentieren, dass die Elemente, die später das komplette Programm ausmachen Bestandteil der Dokumentation sind. Dies führt dazu, dass bei zukünftigen Änderungen an der Dokumentation und/oder dem Quellcode nur eine Quelle maßgeblich ist und bearbeitet werden muss. Dieses Vorgehen ermöglich die Softwareentwicklung von der groben Vorstellung der Problemlösung über die Ausarbeitung von Details gemäß der Top-Down-Entwicklung. Die Darstellung wird mit erläuternden Texten versehen und stellt so die Dokumentation des Problemlösevorgangs vor. Dabei kann durchaus von einem strikten Top-Down-Entwurf abgewichen werden. Aus dieser Darstellung kann automatisch auflauffähiger Quellcode produziert werden.



Willy-Brandt-Gesamtschule, Bergkamen
Fachgruppe Informatik

StD Dipl.-Inform. Dr. L. Humbert — Name:

Informatikgrundkurs 11. Jhg. – Lernzielkontrolle 1 – Gruppe B

1. **Aufgabe (12 Punkte)**
Objektorientierung – Klassen – Objekte – Methoden

Identifizieren Sie in dem nebenstehenden Python-Quellcode die auftretenden Klassen und Objekte und ordnen Sie die in dem Quellcode benutzten Methoden den Objekten/Klassen zu, die diese Methoden »anbieten«. Geben Sie bitte ausschliesslich die Klassen, Objekte und Methoden an, die tatsächlich verwendet werden. Wenn Sie für auftauchende Klassen im Quellcode kein Objekt finden, so ist das nicht als problematisch anzusehen.

Tragen Sie Ihre Ergebnisse in die untenstehende Tabelle ein.

⟨*sumKernTest.py** 1⟩≡
```
from sumkern import Bildschirm
from sumkern import Stift
from sumkern import Maus
from sumkern import Tastatur
meinBildschirm= Bildschirm()
meinStift= Stift()
meineMaus= Maus()
meinStift.bewegeBis(
        maus.hPosition(),maus.vPosition())
stift.zeichneKreis(8)
```

Fehlerhafte Einträge führen zu Punktabzug.

Klasse	Objekt(e)	Methode(n)

2. **Aufgabe (9 Punkte)**
Informatik – Datenschutz – Objektorientierung

a) Geben Sie **Ihre** Definition für Informatik an.
b) Was bedeutet »informationelle Selbstbestimmung«?
c) Grenzen Sie die Begriffe **Klasse** und **Objekt** voneinander ab.

[2004_11_17_IF_GK_11_LZK_1_B]

Dr. L. Humbert, mailto:humbert@willy-brandt.un.nw.schule.de?Subject=Fragenzu2004_11_17_IF_GK_11_LZK_1_B

Arbeitsblatt mit integriertem Quellcode (Noweb)

Für ein neues Projekt sollten Sie sich einmal mit dieser Variante beschäftigen, da gerade in diesem Fall die Stärke der integrierten Dokumenation unmittelbar zu Tage tritt.

Wollen Sie allerdings *nur* die hervorragenden Möglichkeiten des Textsatzes von LaTeX nutzen, um bereits entwickelten Quellcode *schön* darzustellen, so ist das im Folgenden beschriebene Paket geeignet[er].

Das Paket Listings – Programmcode darstellen

Ein anderer Ansatz – Darstellung existierenden Programmcodes – ist mit dem Paket Listings in hervorragender Weise zu realisieren. Auch dieses Paket unterstützt sehr viele Programmiersprachen.

Wettbewerbe

25. Bundeswettbewerb Informatik – mitmachen

(von Dr. Ludger Humbert) Am **1. September 2006** wurden die Aufgaben für die erste Runde des 25. Bundeswettbewerbs Informatik bekanntgegeben: ↘ www.bwinf.de

Bundeswettwerb Informatik – Logo

Wie in den zurückliegenden Jahren auch, ist die Teilnahmen von Gruppen an der ersten Runde erwünscht. Teilnehmen können Jugendliche, die am Tag des Einsendeschlusses der ersten Runde nicht älter als 21 Jahre sind.

Einsendeschluss für die 1. Runde ist der 13. November 2006 Zum dritten Mal wird eine „Junioraufgabe“ jüngeren Schülerinnen und Schülern den Einstieg in den Wettbewerb erleichtern. Die Anmeldung zum 25. BWINF erfolgt mit der Einsendung von Lösungen oder nach dem Wettbewerbsstart unter ↘ www.bwinf.de.

* * *

24. Bundeswettbewerb Informatik – Endrunde

Gerade für Informatiklehrerinnen stellen die Aufgaben in der Endrunde immer wieder eine Quelle für anspruchsvolle und spannende Aufgabenkontexte bereit. Die Aufgaben der Endrunde des 24. Bundeswettbewerb Informatik (und vorgängiger Wettbewerbe) sind über die Webseiten des BW INF zugänglich. Scheuen Sie sich nicht, diese Aufgaben mal in Augenschein zu nehmen und damit Aufgabenideen zu finden, die von Ihren Schülerinnen im Rahmen von Projekten angegangen werden können: ↘ www.bwinf.de

* * *

Ergänzung: LaTeX – Teil 11: ER-Diagramme

Nach dem Erscheinen des letzten Beitrags aus der LaTeX-Serie erhielt ich die E-Mail eines ehemaligen Referendars, der mitteilte, dass er ein LaTeX-Paket zur Erstellung von ER-Diagrammen gefunden hat: ↘ haspe.homeip.net:8080/cgi-bin/pyblosxom.cgi/LaTeX/2006-09-08_hermpic.html

Lesen

(von Dr. Ludger Humbert)

Ausstellung: Modernism …

Die Zuordnung zu der Rubrik Lesen mag zunächst irreführend erscheinen. Ausschließlich durch **Lesen** erschließt sich die Ausstellung, auf die hier hingewiesen wird, kaum. Andererseits bedarf es sehr wohl des **Lesen**s, um die Qualität und die Seiteneffekte, die diese Epoche auf unsere Sicht der Welt hat, zu erkunden.

In der Ausstellung »Modernism: Designing a New World« richtet **MARTa Herford** den Blick auf die Jahre 1914 bis 1939, um die Grundgedanken und Themen der Künstler, Designer, Architekten, Fotografen und Regisseure vorzustellen und zu beleuchten, deren Werke bis heute prägend für die Gestaltung unserer Umwelt sowie unserer Kultur insgesamt sind. [vgl. ↘ www.marta-herford.de/pages/de/museum/aktuelleausstellungen/modernism.html]

Ich bin angetan von den Exponaten der Ausstellung und empfehle sie jedem an Informatik interessierten Menschen. Diese Empfehlung resultiert nicht nur aus dem Exponat, das die Entwicklung der Darstellung des Netzes der *London Underground* zeigt und damit für die Endrunde des 24. BW Inf gut geeignet gewesen wäre. ↘ bwinf.de/uploads/media/bwi24/runde3/graphen.pdf

London Underground Diagram 1933 Quelle: ↘ homepage.ntlworld.com/clive.billson/tubemaps/1933.html

Vielmehr zeigen sehr viele Exponate in meinen Augen eine informatische Sicht auf diverse Bereiche des Lebens, wie sie in der Zeit des »Modernism« entwickelt wurden. Diese aus heutiger Zeit durchaus kritisch bewertete Sicht findet sich zum Beispiel in den Versuchen zur Rationalisierung der Küchenarbeit durch Minimierung der Laufwege zur Erledigung des Zubereitens einer Mahlzeit. Auch in den Exponaten zur Bühnenkunst finden sich überraschend viele Beispiele, die die Präsentation weniger dem Können und Fähigkeiten der Darstellenden überläßt, als vielmehr eine Konstruktion vorgibt, die keinen Platz für Zufälligkeiten erlaubt.

Der Katalog zur Ausstellung wurde – wie die gesamte Ausstellung – in Großbritannien erstellt und liefert auf 447 Seiten einen umfassenden Blick auf die Ausstellung.

- ISBN: 1851774777
- Quanta Costa: 28 Euronen

Da will ich hin – Öffnungszeiten und Preise

Die Ausstellung läuft noch bis zum 7. Januar 2007. Öffnungszeiten: Dienstag bis Sonntag und an Feiertagen: 11.00 bis 18.00 Uhr. Jeden 1. Mittwoch im Monat: 11.00 bis 21.00 Uhr.
9,00 € Eintritt pro Person
5,00 € ermäßigter Eintritt

Abitur in Informatik – Python die Sprache der Wahl

(von Dr. Ludger Humbert)

Bisher erschöpft sich die Unterstützung der Lehrerinnen und damit der Schülerinnen für die Vorbereitung zum Zentralabitur im Wesentlichen in dem Angebot einiger weniger grundlegender Vorgaben, Beispielaufgaben und Schnittstellen. Der Blick in die veröffentlichten Beispielaufgaben ließ im vergangenen Jahr erhebliche Fehler deutlich werden. Diese wurden offensichtlich, als ich Lösungen für die Aufgaben in Python implementieren wollte: ↘ www.die.informatik.uni-siegen.de/pipermail/gi-fg-informatische-bildung-nrw/2005-August/000090.html
Die Lösungen wurden unter ↘ humbert.in.hagen.de/ddi (unten auf der Seite) veröffentlicht.

Die Klasse TList

TList
+__init__() +isEmpty(): Boolean +isBefore(): Boolean +isBehind(): Boolean +next() +previous() +toFirst() +toLast() +getItem(): TObject +update(pObject:TObject) +insertBefore(pObject:TObject) +insertBehind(pObject:TObject) +delete() +destroy()

Klassendiagramm TList nach der Vorlage »datentypen-objekto-ansatz«

Auch im Jahr 2006 (ein Jahr nach meiner dezidierten Kritik) wird weiterhin als Beispiel von offizieller Seite ↘ www.learn-line.nrw.de/angebote/abitur-gost/download/if-datentypen-objekto-ansatz-delphi.pdf die Klasse **TList** (Seite 9–11) angegeben. Wer erinnert sich noch, woher das »T« kommt? Es wurde vor langer Zeit für Implementierungen in Pascal eingeführt, um selbstdeklarierte Typen(!) zu bezeichnen – müsste jetzt nicht »K« für Klasse gewählt werden –?
Da z. B. Standarddatentypen nicht über ein Präfix verfügen, sollten sich die Autoren m. E. von dieser überholten und didaktisch umstrittenen Präfixnotation verabschieden. Eher sollte über eine vereinbarte Schreibweise (Gross für Klassen - klein für Objekte) sichergestellt werden, was eine Klasse ist, denn über ggf. mehrdeutig interpretierbare Präfixe.

Es wird eine Klasse angegeben, die mit **vierzehn** Methoden ausgestattet ist. Übliche, d. h. in der *Informatik* bekannte Schnittstellen zur Listenverwaltung enthalten (je nach Implementierung) ca. **sechs** Methoden, um den notwendigen Funktionsumfang zur Verfügung zu stellen.

Ich fragte die Autoren öffentlich, wie sie auf diese Schnittstelle gestossen sind. Ich erhielt auf meine Anfrage ↘ www.die.informatik.uni-siegen.de/pipermail/gi-fg-informatische-bildung-nrw/2005-August/000090.html keine Antwort.

Sobald Klassen deklariert werden, sollte dafür Sorge getragen werden, dass sie **minimal** festgelegt sind.

Es ist jeder informatischen Strukturierung abträglich, wenn (wie in diesem Fall) mehrere Methoden zur Erledigung einer Aufgabe verwendet werden können. So etwas sollte nicht veröffentlicht werden (selbst wenn es Bestandteil einer Klassenbibliothek ist).
Erst recht kann eine solche Schnittstelle nicht zum Vorbild erklärt werden. Jedes von mir konsultierte Standardwerk der Informatik zeigt auf, wie die Datenstruktur Liste implementiert werden kann, ohne einen derartigen Wildwuchs an Methoden bereitzustellen.

Die Klasse ListMinimal

Ich schlage vor, statt dieser Schnittstelle zum Beispiel die folgende Schnittstelle zu verwenden.

ListMinimal
+__init__() +next() +first() +insert(object:Element) +delete(object:Element) +append(object:Element) +seek(object:Element): Element

Element
+key +zeiger: Element
+__init__(key)

Eine [relativ] minimale Klasse zur Nutzung von Listen

Sie findet sich in der Standardliteratur an Stellen, die sich mit der effizienten Implementierung von Listenstrukturen beschäftigen. Darüber hinaus wird üblicherweise in der deutschsprachigen Informatikliteratur mit deutschen Bezeichnern gearbeitet, wenn Datenstrukturen beschrieben werden. Als Beispiel sei auf die Standardeinführung in die Informatik von Helmut Balzert verwiesen. Im einführenden »Lehrbuch Grundlagen der Informatik« wird in Kapitel 17 ausführlich zu Listen gearbeitet.
Auch Balzert scheut sich nicht, auf andere einführende Werke zu verweisen und gerade bei der Listendarstellung differenziert auf die Seiteneffekte einer an der effizienten Implementierung orientierten Datenstruktur hinzuweisen.

ListeBalzert
+anfang +ende +vorgaengerAktuellerZeiger
+__init__(anfang) +naechstesElement(): Element +setzeAktuellerZeigerZurueck() +einfuegenElement(object:Element) +entfernenElement(object:Element) +anfuegenElement(object:Element) +suchenElement(object:Element): Zeiger

Eine minimale Klasse zur Nutzung von Listen vgl. Balzert

Maßgabe **jeder** informatischen Modellierung muss eine möglichst klare und verständliche Schnittstelle sein. Gerade für häufig benutzte Schnittstellen werden darüber hinaus gewisse Anforderungen an die Effizienz und Eleganz der Implementierung gestellt.

Weitere fachliche Kritik

In dem o. g. Dokument wird durchgängig ein Begriff benutzt, der **fachlich falsch** ist: »Benutzerschnittstelle« (ab Seite 30ff). Vor ca. 20 Jahren wurde in der Informatik erkannt, dass dieser Begriff durch **Benutzungsschnittstelle** zu ersetzen ist. Es ist sehr ärgerlich, wenn solche Fehlleistungen öffentlich publiziert werden.

Ist Pascal [noch] geeignet?

Wenn Sie sich bis zu dieser Frage vorgearbeitet haben und diesen Artikel nicht als beleidigend empfinden, werden Sie sich gewiss die in der Zwischenüberschrift gestellte Frage gestellt haben. Das zentrale Problem besteht darin, dass Pascal eine stark typisierte Sprache ist und nicht erlaubt, dass im Nachhinein Daten eines anderen Typs verwendet werden, als bei der Deklaration festgelegt. Daraus resultiert, dass alle Elemente, die in einem Graphen, einer Liste, etc. organisert werden, letztlich von einer vorher festgelegten Klassen abzuleiten sind. Da die Mehrfachvererbung nicht erlaubt ist, schränkt diese Festlegung aller streng typisierten Sprachen die Möglichkeiten, beliebige Objekte in einer solchen Datenstruktur zu organisieren, ganz erheblich ein.
Hier bieten schwach typisierte Sprachen einen grossen Vorteil, da sich jederzeit beliebige Strukturen miteinander verzahnen lassen. Dennoch kann an kritischen Stellen über eine Typprüfung erreicht werden, dass die Typsicherheit (wenn sie gefordert ist) gewährleistet werden kann.

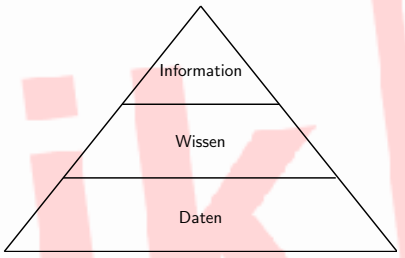
Die nächste Frage betrifft die Möglichkeit, vordefinierte Typen zu erweitern. Dies ist in streng typisierten Sprache üblicherweise nicht vorgesehen. Dabei wird deutlich, dass Mischsprachen, die imperativen Ursprungs sind, für die objektorientierte Implementierung aus didaktischer Sicht nicht gut geeignet sind, da sie mit Restriktionen aufwarten, die aus objektorientierter Sicht mit nicht orthogonalen Einschränkungen verbunden sind. Das Orthogonalitätskriterium ist für die didaktische Bewertung der Eignung einer Sprache ein hartes Kriterium und sollte nicht ohne schwerwiegende Gründe mißachtet werden.

Glossar: Information

(von Dr. Ludger Humbert)

Information

»Der Begriff der Information kristallisiert sich seit einigen Jahrzehnten als ein ähnlich grundlegender Begriff heraus, wie das die Begriffe Energie und Materie schon seit langem als Basis jeder Naturwissenschaft geworden sind.« [Quelle: ↘ de.wikibooks.org/wiki/Über_das_Wesen_der_Information] Soweit stimme ich mit den Hinweisen der Quelle überein. Eine Definition des Begriffs ist nicht einfach. Dies liegt in dem Unterbau begründet, der begrifflich vorausgesetzt werden kann. In der Informatik können wir eine Schichtung der drei Begriffe Daten, Wissen, Information ausmachen. Dabei ist die Reihenfolge der Begriffe Wissen und Information nicht einheitlich. Hier neige ich der oben angegebenen Reihenfolge zu, betrachte also Information als den umfassenden Begriff.



Daten Wissen Information

Mit der Shannon'schen **»Informations«**theorie wurde die Einschränkung des Begriffs auf eine quantifizierbare Eigenschaft von Nachrichten (Daten) vorgenommen, die einem modernen Verständnis nicht mehr entspricht. Daher wird heute von der einengenden Sicht auf den Informationsgehalt in weiten Teilen der Informatik abgesehen.

Es gibt viele Varianten, in denen der Begriff in der Informatik benutzt wird – dabei wird häufig Information synonym für Daten benutzt. Dies macht eine fachsprachlich korrekte Einordnung nicht einfach. In der Informationswissenschaft wird die begriffliche Schärfe unabdingbar und so wurde der Informationsbegriff so geschärft, das er gegenüber Wissen und Daten sorgfältig abgegrenzt wird.

Man ordnet die drei Begriffe wohlbekannten Fachbegriffen zu: Syntax (Daten), Semantik (Wisssen) und Pragmatik (Information). Mit Hilfe von Beispielen werden die drei Qualitäten illustriert [vgl. Humbert: Didaktik der Informatik, 2. Aufl., S. 29]:

Syntax: Ein Dokument wird als Folge von Zeichen/Symbolen aufgefasst. Auf dieser Ebene kann beispielsweise mit Methoden agiert werden, die Zeichenketten in Texten oder die nach Merkmalen wie Farbe, Textur und Kontur in Bilddaten suchen.
Semantik: Bedeutung eines Dokumentes. Eine Methode auf dieser Ebene ist die Suche nach bedeutungstragenden Elementen in einem Textdokument oder die Suche nach Bildern, die bestimmte (Arten von) Objekten enthalten (Menschen, Häuser, Autos, . . .).
Pragmatik: Zweckorientierte Nutzung eines Dokumentes. Zum Beispiel sucht eine Referendarin Literatur zur einem bestimmten Thema. Bildarchive werden häufig von Journalistinnen in Anspruch genommen, um einen Artikel zu illustrieren; dabei ist meist das Thema vorgegeben, aber nicht der Bildinhalt.