Technischer Report

Algorithmen und Datenstrukturen

Foliensatz, 01.12.05
wird noch ergänzt!

Wintersemester 2005/2006

Werte

Zahlenwerte

- Zahlen in der Informatik
  - Eigenschaften ähnlich wie in der Mathematik
    - *Informatische Zahlen sind ein Modell der mathematischen*
  - **Ganze Zahlen** (*integer*:)
    - …, –2, –1, 0, 1, 2, …
    - Teilmengen hiervon
      - {0, 1, 2, ..., MAX_VALUE} – CARDINALs in Modula2
    - Meiste kleinste und größte Zahl
      - Integer.MAX_VALUE, Integer.MIN_VALUE
    - Ausnahmen
      - ASN.1
      - BigInteger
    - gleiche Genauigkeit wie in der Mathematik!

Zahlenwerte

- Dezimalzahlen
  - Dezimalpunkt
    - statt Dezimalkomma, wie im Deutschen üblich, Punkt
    - in der Informatik weitgehend durchgesetzt
    - In vielen Anwendungen auch Komma (Lokalisierung!)
  - Dezimalpunkt trennt ganzen Zahlenwerte von Dezimalwert
    - 3.5: drei ganzen Einheiten + Fünffache einer zehntel Einheit (also 0.5)

Zahlenwerte

- Dezimalzahlen
  - In Physik und Technik übliche Darstellung
    - $3.5 = 35 \cdot 10^{-1}$
      - 35 mit Faktor $10^{-1}=0.1$ multiplizieren
        - Faktor $10^{0}=1$ wird in der Regel fortgelassen
      - –1 im Exponenten von 10 gibt Stelle des Dezimalpunkts an
  - in der Informatik ähnliche Darstellung
    Gleitkommazahl (*floating point; float*, *real*, *double*)
    - Wert für Dezimalstellen
      - Mantisse
    - Wert für Kommastelle
      - Exponent
    - Ex (ex) statt 10^{x}
      - 35 ≡ 35.0
      - 35.5 ≡ 0.355e2
      - 34567 ≡ 3.4567E4
      - 0,00034 ≡ 34e-5

Zahlenwerte

- Darstellung ganzer Zahlen
  - von der verwendeten Maschine abhängig
    - **`java.nio.ByteOrder.nativeOrder()`**
    - **`java.nio.ByteOrder.LITTLE_ENDIAN`**
    - **`java.nio.ByteOrder.BIG_ENDIAN`**
  - Dualzahlenwert
    - Folge von Dualstellen mit Zifferwerten 0, 1
      - Ziffer an Stelle k (von rechts gezählt) hat Wert 2^{k-1}.
      - 1. Stelle: $2^0=1$
      - 2. Stelle: $2^1=2$
      - 3. Stelle: $2^2=4$
      - 4. Stelle: $2^3=8$

Zahlenwerte

- Darstellung ganzer Zahlen
  - Vorzeichen
    - Häufig gibt letzte Stelle (erste Stelle von links) das Vorzeichen an
    - Verschiedene Möglichkeiten
    - Einerkomplement
      - z (00110011), $-z$ (11001100)
      - null (0000), –null (1111)
  - Zweierkomplement
    - z (00110011), $-z$ (11001101)
    - $-_2 z = 2^N -_d z$.
      - $-_2$: Operator in Zweierkomplement
      - $-_d$: Operator im Dualsystem
      - N: Anzahl der Dualstellen
  - N=4

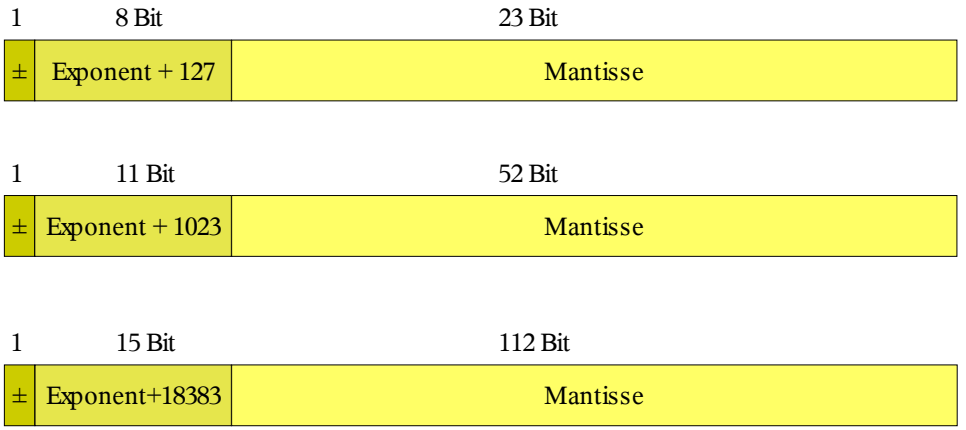
Dualzahl	0	1	10	11	100	101	110	111
Dezimalwert	0	1	2	3	4	5	6	7
Dualzahl	1000	1001	1010	1011	1100	1101	1110	1111
Dezimalwert	-8	-7	-6	-5	-4	-3	-2	-1

Zahlenwerte

- Darstellung ganzer Zahlen
  - $w +_2 (-z) = w +_d (2^N -_d z) = 2^N +_d (w -_d z)$
    - N=4
    - $3 +_2 2 = 3 +_d 2 = 5$
    - $3 +_2 \text{-}2 = 3 +_d (16-2) = 17 = (16) + 1$
    - $3 +_2 -4 = 3 +_d (16-4) = 15 = (16) - 1$
    - $-3 +_2 -2 = (16-3) +_d (16-2) = 27 = (16) + 11$
  - Ein Rechenwerk für alle Operationen!
    - Dual-Rechnung
    - Rechnen mit vorzeichenbehafteten Zahlen
  - Abstraktionsprinzip!

Zahlenwerte

- Darstellung von Gleitpunktzahlen
  - Vorzeichen an linker Stelle
  - Exponenten in geeigneter Normaldarstellung
  - Mantisse in normalisierten Form 0.1xyz…
- Standard ANSI/IEEE 754–1985



Zahlenwerte

- Darstellung von Gleitpunktzahlen
  - Zahlenbereich
    - Spanne zwischen kleinster und größter Zahl
      - Zahlenbereichsüberschreitung
        - Zahlenüberlauf
        - *underflow*
        - *overflow*
    - Zahlenwert unendlich: 1/0.0
      - POSITIVE_INFINITY
      - NEGATIVE_INFINITY
      - isInfinite()
    - Undefinierte Zahlenwerte
      0/0, POSITIVE_INFINITY+NEGATIVE_INFINITY
      - NaN
      - Kann nicht auf Gleichheit abgefragt werden!
      - isNaN()

Zuweisung von Zahlenwerten

- Zuweisung von Zahlenwerten
  - Variablen einen Wert zuweisen
    - `int Wert, GanzeZahl;`
    - `double Zahl;`
    - Definition (definition)
    - Deklaration (declaration)
  - Wertzuweiung
    - `adr Wert := con 27`
    - `adr Zahl := con 199.5`
    - `adr GanzeZahl := con 3.5`
  - Unterschiedliche Bedeutung
    - Speichersemantik
    - Wertesemantik
      - Rundung (*round*)
      - Abschneiden (*truncate*)

Zuweisung von Zahlenwerten

- Zuweisung von Zahlenwerten
  - Unterschiedliche Bedeutung
    - Speichersemantik
    - Wertesemantik
      - Rundung (*round*)
      - Abschneiden (*truncate*)
      - `adr GLPZahl     :=  con 3.5E4`
      - `adr GanzeZahl   :=  val GLPZahl`
      - `adr GLPZahl     :=  con 3.49`
      - `adr GanzeZahl1  :=  val GLPZahl`
      - `adr GLPZahl     :=  con 3.51`
      - `adr GanzeZahl1  :=  val GLPZahl`
      - `adr GLPZahl     :=  con 3.5`
      - `adr GanzeZahl   :=  val GLPZahl`

Zuweisung von Zahlenwerten

- Zuweisung von Zahlenwerten
  - Typanpassung oder -umwandlung (type conversion)
    - Umwandlung eines Zahlobjekts eines Typs in anderen Typ
      - z.B. Gleitkommazahl in eine ganze Zahl
  - Wert soll erhalten bleiben
    - implizite Typumwandlung
    - explizite Typumwandlung
      - (integer) GLPZahl
      - integer(GLPZahl)
      - real2integer(GLPZahl)
      - (int) GLPZahl
      - Java: (TypBezeichner)<Ausdruck>
    - type casting
      - verschiedene Bedeutungen!

Zuweisung von Zahlenwerten

- Dualzahl, Binärzahl
  - binär (binary)
    - zweiwertiges Objekt
- BCD-Zahlen (binary coded decimals)
  - Dezimalziffer (0, 1,… 9) mit vier Bits (0000, 0001,… 1001)
  - erspart Umrechnung zwischen Programmdarstellung und Rechnerdarstellung
  - heute selten
  - Nutzen sehr gering ist; wiegt Nachteile nicht auf
- Konstante (Literale)
  - `final double Pi = 3.14159265;`
  - `final String Hallo = "Dies ist ein Text";`

Logische Werte

- Logische Werte
  - Boolsche Werte (George Boole)
  - zweiwertiges System
    - wahr (*true*)
    - falsch (*false*)
    - Schlüsselwörter
      - **`boolean`**
        - **`true`**
        - **`false`**
    - andere Darstellungen möglich
    - keine expliziten Werte (COBOL, C, C++)

Logische Werte

- Logische Werte
  - `boolean  istWahr = true, istGleich = false;`
  - `istWahr = ! istGleich; // Negation: not`
  - Bedingender Junktor
    Wenn ausreichend, wird nur ein Operand ausgewertet
    - `if(istWahr && istGleich) ..// Konjunktion: and`
    - `if(istWahr || istGleich) ..// Disjunktion: or`
  - Auswertender Junktor
    Beide Operanden werden ausgewertet
    - `if(istWahr & istGleich) ... // Konjunktion: and`
    - `if(istWahr | istGleich) ... // Disjunktion: or`
    - `if(istWahr ^ istGleich) ... // Exklusives Oder`

Texte

	0		1		2		3		4		5		6		7	
0	NUL	0	DEL	16	SP	32	0	48	@	64	P	80	`	96	p	112
1	SOH	1	CC1	17	!	33	1	49	A	65	Q	81	a	97	q	113
2	STX	2	CC2	18	"	34	2	50	B	66	R	82	b	98	r	114
3	ETX	3	CC3	19	#	35	3	51	C	67	S	83	c	99	s	115
4	EOT	4	CC4	20	$	36	4	52	D	68	T	84	d	100	t	116
5	ENQ	5	NAK	21	%	37	5	53	E	69	U	85	e	101	u	117
6	ACK	6	SYN	22	&	38	6	54	F	70	V	86	f	102	v	118
7	BEL	7	ETB	23		39	7	55	G	71	W	87	g	103	w	119
8	BS	8	CAN	24	(	40	8	56	H	72	X	88	h	104	x	120
9	HT	9	EM	25	)	41	9	57	I	73	Y	89	i	105	y	121
10	LF	10	SUB	26	*	42	:	58	J	74	Z	90	j	106	z	122
11	VT	11	ESC	27	+	43	;	59	K	75	[	91	k	107	{	123
12	FF	12	FS	28	,	44	<	60	L	76	\	92	l	108	|	124
13	CR	13	QS	29	-	45	=	61	M	77	]	93	m	109	}	125
14	SO	14	RS	30	.	46	>	62	N	78	^	94	n	110	~	126
15	SI	15	US	31	/	47	?	63	O	79	_	95	o	111	DE	127

- Texte
  - Folgen von Zeichen sind Texte
  - ASCII-Alphabet

Texte

- Unicode
  - 16 Bits
  - In Java mittels Fluchtsymbol dargestellt: '\uwxyz'
    - Ab 900 (dezimal) findet man die Zeichen
      - ΄΅Ά·ΈΉΊ Ό ΎΏΐΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡ ΣΤΥΦΧΨΩΪΫάέήίΰαβγδεζηθικλμνξοπρςστυφχψωϊϋόύώ
    - Ab 1015 (dezimal) findet man die Zeichen
      - АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ абвгдежзийклмнопрстуфхцчшщъыьэюя
    - Ab 1475 (dezimal) findet man die Zeichen:
      - תשרקצץפףעסנןמלם ... ״ווויי ... כךיטחזהוגדבא
    - Ab 1565 (dezimal) findet man die Zeichen:
      - ۃابةتثجحخدذرزسشصضطظعغ ... ـفقكلمنهوىيٍّ

Texte

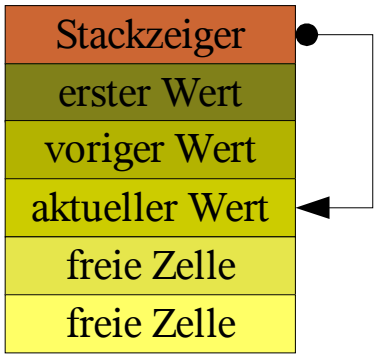
- Texte in Java
  - `String Text = "Zahl = ";`
    - `int Zahl = 123;`
    - `String NeuerText = Text + Zahl;`
      `// NeuerText == "Zahl = 123"`
    - `Text.length()`
    - `Text.substring(2,4) == "hl"`
  - `StringBuffer sb = new StringBuffer();`
    - `sb.append("Anhängen");`
    - `sb.insert("Einfügen");`
    - `sb.length();`
    - `sb.capacity();`
    - `sb.substring(..);`
    - `sb.charAt(4);`
    - `sb.toString();`

Speicheradressen

- Speicheradressen
  - Adresse (eines Objekts) im Speicher
  - kann im Prinzip beliebig manipuliert werden
    - Kopieren, Setzen
    - Addieren (Inkrementieren), Subtrahieren (Dekrementieren)
    - Multiplizieren, Dividieren
    - **`adr val SprungAdresse`**
    - **`val val ObjektAdresse`**
    - **`val val StackAdresse`**
  - Zeiger
  - Pointer (C, C++)
  - Reference
  - Adressvariable

Stack

- Stack (Stapel)
  - Daten werden in gleicher Reihenfolge eingefügt wie entfernt
  - push datum
    - füge datum in Stack ein
  - pull (pop)
    - entferne zuletzt eingefügtes Datum
    - pull kann auch Wert liefern (getPull adr wert)
  - meist als unbeschränktes Feld implementiert
    - unbeschränkt heißt in der Informatik
      - 'Betriebssystem' stellt Speicher zur Verfügung
      - wenn nicht ausreichend wird Speicher nachgereicht
      - bis Hauptspeicher erschöpft
      - Stack ist immer endlich
        - und manchmal eine kritische Ressource



Felder

- Felder (*array*, *field*)
  - Folge von Daten gleichen Typs
    - Fak[0]=1, Fak[1]=1, Fak[2]=2, Fak[3]=6, Fak[4]=24,
    - Textur[0] = "images\\brick1.jpg",
      Textur[1] = "images\\background1.jpg"
  - Funktion mit endlichem Definitionsbereich
    - Histo[i] ist Anzahl von Studenten mit [i*5..(i+1)*5] Punkten
  - Vektor mit endlicher 'Dimension'
    - (x,y) Koordinaten auf dem Bildschirm
    - (x,y,z,w) homogene Koordinaten in OpenGL
    - (x,y,z,t) Minkowkische Raum-Zeit-Welt

Felder

- Felder (array, field)
  - Notation
    - FeldName[12] (Indexnotation: f_{12})
    - FeldName(12) (Funktionsnotation: $f(12)$)
  - Deklaration
    - declare integer array FeldName[1..100]
      - FeldName[12] ergibt das zwölfte Element
    - declare integer array FeldName[0..100]
      - FeldName[12] ergibt das 13. Element
    - In Java
      - `int [] FeldName;`
      - `int FeldName [];`
      - `int FeldName [] = new int[100]`
        `FeldName[0], FeldName[1], ... FeldName[99].`

Datensätze

- Datensätze
  - Record (record), structure
    - Verbund
      - Daten zu einem Objekt zusammengefasst
      - Typ der Daten beliebig
        - Zahlen
        - Zeichen
        - Texte
        - Felder
      - Zugriff über eine Adresse
    - In Java: **`class`** als Typbeschreibung
      - ```
        class RecordName {
          int GanzeZahl = 3;
          char Zeichen = 'c';
          String Name = "Emil";
        }
        ```
        ```

Datensätze

- Datensätze
  - In Java: **`class`** als Typbeschreibung
    - ```
      class RecordName {
        int GanzeZahl = 3;
        char Zeichen = 'c';
        String Name = "Emil";
      }
      ```
  - Object durch Instanzierung einer Klasse:
    - ```
      RecordName object = new RecordName();
      ```
  - Zugriff durch 'Dereferenzierung'
    - ```
      Recname.GanzeZahl := 44;
      ```
    - ```
      Recname.Zeichen := 'A';
      ```
    - ```
      Recname.Name := "Tim";
      ```
      ```

Datensätze

- Datensätze unterstützen Zugriffskontrolle
  - Alternative
    - Daten in einzelne Variablen oder Felder unterbringen
    - Zuordnung nur schwierig erkennbar
  - nichtnumerische Datenverarbeitung
    - Administration großer Datenmengen
      - Betriebswirtschaftliche Datenverarbeitung
      - Administration

Datensätze

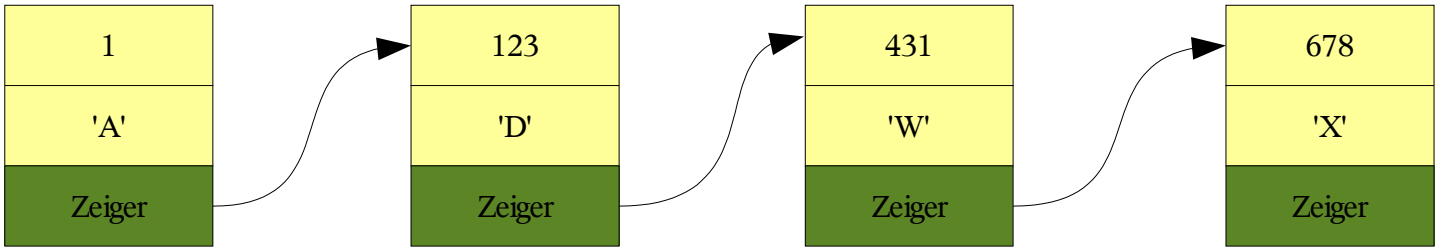
- Jedes Objekt durch Position im Record festgelegt
  - Beispiel: komplexe Zahlen
    - ```
      class Complex {
        double real, // real number
        double imag; // imaginary number
      }
      Complex zahlA = Complex();
      Complex zahlB = Complex();
      ```
  - Anordnung im Speicher
    - ```
      42 zahlA      con 2.0 // real
      ```
    - ```
      43            con 3.0 // imag
      ```
    - ```
      44 zahlB      con 1.0
      ```
    - ```
      45            con 5.0
      ```
      ```

Datensätze

- Zugriff auf Objekte der Datensätze
  - ```
    ZahlA -> real := 2.0; // C, C++
    ZahlA -> imag := 3.0;
    ZahlB -> real := 1.0;
    ZahlB -> imag := 5.0;
    ```
  - ```
    ZahlA^.real := 2.0;   // Pascal
    ZahlA^.imag := 3.0;
    ZahlB^.real := 1.0;
    ZahlB^.imag := 5.0;
    ```
  - ```
    ZahlA.real := 2.0;    // Java
    ZahlA.imag := 3.0;
    ZahlB.real := 1.0;
    ZahlB.imag := 5.0;
    ```
    ```

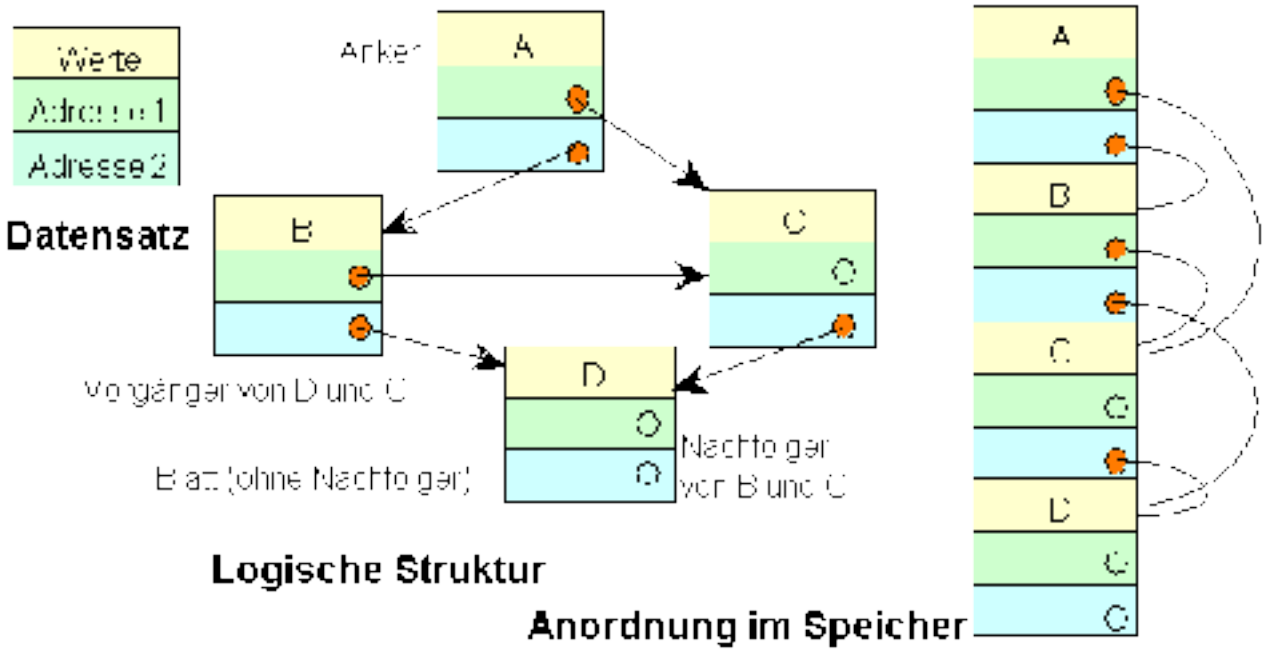
Datensätze

- Records können auch Referenzen auf Records enthalten
  - Listen,
  - Bäume
  - Allgemeine Datenstrukturen



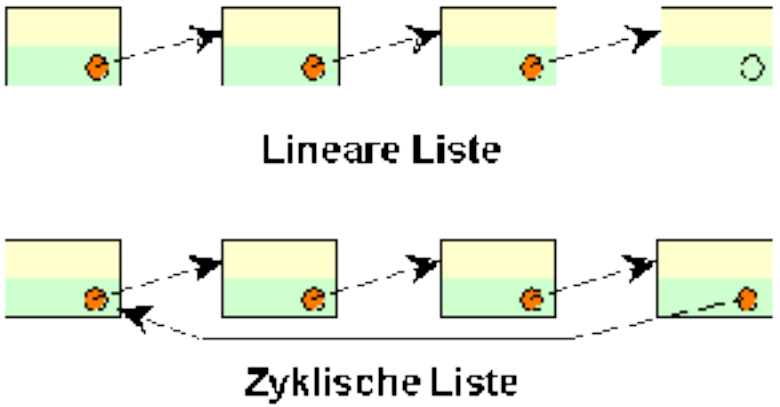
Datensätze

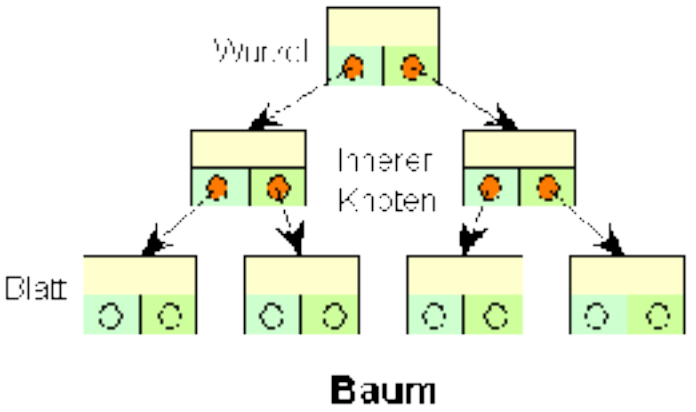
- Referenzen auf Records in Records
  - Listen,
  - Bäume
  - Allgemeine Datenstrukturen



Datensätze

- Allgemeine Datenstrukturen





Datensätze

- Listen in LISP