4. Number Representations

Domain of Types **`int`**, **`float`** and **`double`** Mixed Expressions and Conversion; Holes in the Domain;Floating Point Number Systems; IEEE Standard; Limits of Floating Point Arithmetics; Floating Point Guidelines

Educational Objectives

- You have a good understanding how a computer represents numbers.
- You can transform integers in **binary representation** and perform computations.
- You understand how the value range of integers is chosen.
- You can describe the representation of floating-point numbers in general.
- You know the three **floating-point rules**.
- You can use booleans and **boolean expressions** in Java.

Binary Number Representations

Binary representation (Bits from $\{0, 1\}$)

$$b_n b_{n-1} \dots b_1 b_0$$

corresponds to the number $b_n \cdot 2^n + \dots + b_1 \cdot 2 + b_0$

101011 corresponds to **43**.

Least Significant Bit (LSB)

Most Significant Bit (MSB)

Binary Numbers: Numbers of the Computer?

Truth: Computers calculate using binary numbers.

NEUE ZÜRCHER ZEITUNG

TECHNIK

Mittwoch, 30. August 1939 · Blatt 15 · Mittagausgabe · Nr. 2790 (30)

Das programmgesteuerte Rechengerät an der Eidgenössischen Technischen Hochschule in Zürich

Die Entwicklung programmgesteuerter Rechenmaschinen in den Vereinigten Staaten von Amerika wurde in den Artikeln „Elektronische Rechenmaschinen" (vgl. Nr. 2140 der „N. Z. Z." vom 15. Oktober 1948) und „Die neueste elektronische Rechenmaschine" (vgl. Nr. 871 der „N. Z. Z." vom 26. April 1950) behandelt. Nachstehend soll von einem Gerät deutscher Herkunft — Zuse K.-G., Neukirchen — die Rede sein, welches im Juli dieses Jahres am Institut für angewandte Mathematik der Eidgenössischen Technischen Hochschule in Zürich, das unter der Leitung von Prof. Dr. E. Stiefel steht, in Betrieb genommen wurde. Damit ist diesem Institut in der Lage, dem in der Schweiz immer stärker werdenden Bedürfnis nach einer leistungsfähigen Zentralstelle für numerische Rechnungen wenigstens teilweise gerecht zu werden. Bereits sind einige mathematische Probleme behandelt worden, und die Erledigung vieler anderer Aufgaben ist vorbereitet.

Merkmale des Gerätes

Das Gerät ist ein Glied in dem längeren Entwicklungsprogramm des Ingenieurs *Konrad Zuse*; es wurde im Auftrag des *Institutes für angewandte Mathematik* der E. T. H. unter Berücksichtigung von dessen Wünschen und Ideen von Zuse als „*Modell Z 4*" konstruiert. Die ursprüngliche Entwicklung in Deutschland erfolgte in den Kriegsjahren und verlief völlig unabhängig von den Rechenmaschinen in den Vereinigten Staaten. Es ist überaus interessant festzustellen, wie für die meisten wichtigen funktionellen Probleme übereinstimmend gerade dieselbe Lösung gefunden wurde, wie aber andrerseits gewissen Fragen sekundärer Wichtigkeit eine ganz unterschiedliche Bedeutung beigemessen wurde.

Eine kurze technische Charakterisierung lautet wie folgt: Elektromechanisch arbeitendes Gerät mit 2200 Relais, 21 Schrittschaltern und einem Speicher für 64 Zahlen, welcher mit neuartigen, mechanischen Schaltgliedern arbeitet; Verwendung des Dualsystems und der halblogarithmischen Darstellung; Multiplikationszeit 2,5 Sekunden; Programmsteuerung mit Hilfe zweier Lochstreifen, auf die wahlweise umgeschaltet werden kann; Eingabe von Zahlen durch eine Tastatur oder durch einen Lochstreifen; Abgabe der Resultate durch Lampenfeld, Lochstreifen oder Druckwerk.

Lesen wir eine Dezimalzahl von rechts nach links, so erhöht sich das Gewicht von Stelle zu Stelle um den Faktor 10. Im *Dualsystem* ist nun einfach dieser Faktor 10 durch 2 zu ersetzen. Also bedeutet die (nunmehr duale) Zahl abcd,efg den Ausdruck:

$$a \cdot 2^3 + b \cdot 2^2 + c \cdot 2^1 + d \cdot 2^0 + e \cdot 2^{-1} + f \cdot 2^{-2} + g \cdot 2^{-3}$$

Die Zahl 1 wird in beiden Systemen gleich dargestellt. Um jedoch diese von derjenigen Zahlen deutlich zu trennen, schreiben wir die duale 1 als L. — Dagegen weicht schon die 2 ab, indem sie dual L0 lautet; denn dies bedeutet ja $1 \cdot 2^1 + 0 \cdot 2^0 = 2$. Wenn einer Zahl (ohne Stellen nach dem Komma) rechts eine Null zugefügt wird, so vergrößert sie sich um den Faktor 2 (und steht, wie im Dezimalsystem, um den Faktor 10). Auf diese Weise kann aus L0 = 2 auf einfachste Weise gebildet werden: L00 = 4, L000 = 8, L0000 = 16, usw.

Die Dualzahl L0L0L bedeutet nun also:

$$1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 21$$

Ganz analog sind etwaige Stellen nach dem Komma zu interpretieren; so wird L, 0LL wie folgt übersetzt:

$$1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2} + 1 \cdot 2^{-3} = 1 + 1/4 + 1/8 = 1{,}375$$

Änderung des Maßstabes durchgerechnet werden können.

Die beschriebene Darstellung bringt eine gewisse Komplikation der Rechenoperationen mit sich. So müssen vor einer Addition die beiden Summanden zunächst so verschoben werden, daß ihre Kommata untereinander zu liegen kommen, was on Hand eines Beispiels erläutert werden soll. Damit der Leser nicht durch das ungewohnte, duale Zahlsystem verwirrt wird, ist das Beispiel im Dezimalsystem durchgeführt; doch wird dann erinnert, daß das Gerät in Wirklichkeit mit dualen Zahlen rechnet.

Es soll also etwa addiert werden: $2{,}345078 \times 10^5 + 9{,}876543 \times 10^1$ (Man beachte, daß die exponentiale Zahl stets zwischen 1 und 10 liegt, also das Komma nach der ersten Stelle hat.) Nun müssen die beiden Summanden „ausgerichtet" werden, d. h. die beiden Exponenten sind einander gleich zu machen, und zwar erhält der kleinere Exponent den Wert des größeren, also 2. Die Zahlen lauten nun, richtig untereinander geschrieben und addiert, wie folgt:

$$\begin{array}{l} 2{,}345078 \times 10^5 \\ 0{,}009876 \times 10^5 \\ \hline 2{,}355054 \times 10^5 \end{array}$$

Es ist ersichtlich, daß bei der kleineren der beiden

Abb. 2. Der Schaltpult bei der Fertigung eines Rechengliedes. Die Abtaster für den Lochstreifen sind deutlich sichtbar.

Binary Numbers: Numbers of the Computer?

Stereotype: computers are talking 0/1 gibberish

Computing Tricks

- Estimate the orders of magnitude of powers of two.[2]:

$$2^{10} = 1024 = 1\text{Ki} \approx 10^3.$$
$$2^{20} = 1\text{Mi} \approx 10^6,$$
$$2^{30} = 1\text{Gi} \approx 10^9,$$
$$2^{32} = 4 \cdot (1024)^3 = 4\text{Gi} \approx 4 \cdot 10^9.$$
$$2^{64} = 16\text{Ei} \approx 16 \cdot 10^{18}.$$

[2] Decimal vs. binary units: MB - Megabyte vs. MiB - Megabibyte (etc.)
kilo (K, Ki) – mega (M, Mi) – giga (G, Gi) – tera(T, Ti) – peta(P, Pi) – exa (E, Ei)

Definition: Domain

For numeric types the domain defines the numeric interval a the type can cover.

Book on page 24

Domain of Type `int`

```
public class Main {
    public static void main(String[] args) {
        Out.print("Minimum int value is ");
        Out.println(Integer.MIN_VALUE);
        Out.print("Maximum int value is ");
        Out.println(Integer.MAX_VALUE);
    }
}
```

```
Minimum int value is -2147483648.
Maximum int value is 2147483647.
```

Where do these numbers come from?

Domain of the Type `int`

Representation with 32 bits. Domain comprises the 2^{32} integers:

$$\{-2^{31}, -2^{31}+1, \ldots, -1, 0, 1, \ldots, 2^{31}-2, 2^{31}-1\}$$

Negative Numbers (3 Digits)

	a	$-a$	
0	000	000	0
1	001	**111**	-1
2	010	**110**	-2
3	011	**101**	-3
4	**100**	**100**	-4
5	**101**		
6	**110**		
7	**111**		

The most significant bit decides about the sign.

Over- and Underflow

- Arithmetic operations (`+`,`-`,`*`) can lead to numbers outside the valid domain.
- Results can be incorrect!

  `power8`: $15^8 = -1732076671$ `power20`: $3^{20} = -808182895$

- There is **no error message!**

Definition: Floating Point Numbers

*Floating point numbers represent numbers from $\mathbb{R}$ with a fixed number of **significant** number of digits, multiplied by a decimal power (base 10).*

Book on page 67

“Proper Calculation”

```
public class Main {
    public static void main(String[] args) {
        Out.print("Celsius: ");
        int celsius = In.readInt();
        int fahrenheit = 9 * celsius / 5 + 32;
        Out.print(celsius + " degrees Celsius are ");
        Out.println(fahrenheit + " degrees Fahrenheit");
    }
}
```

```
28 degrees Celsius are 82 degrees Fahrenheit.
```

richtig wäre 82.4

Types `float` and `double`

- are the fundamental types for floating point numbers
- approximate the field of real numbers $(\mathbb{R}, +, \times)$ from mathematics
- have a great domain, sufficient for many applications (**`double`** provides more places than **`float`**)
- are fast on many computers

Fixpoint numbers

- fixed number of integer places (e.g. 7)
- fixed number of decimal places (e.g. 3)

```
    0.0824 = 0000000.082
```
← third place truncated

Nachteile

- Domain is getting *even* smaller than for integers.
- If a number can be represented depends on the position of the comma.

Floating point Numbers

- fixed number of significant places (e.g. 10)
- plus position of the comma

$$\mathtt{82.4} = 824 \cdot 10^{-1}$$
$$\mathtt{0.0824} = 824 \cdot 10^{-4}$$

- Zahl ist $\mathit{Mantissa} \times 10^{\mathit{Exponent}}$

Domain

Integer Types:

- Over- and Underflow relatively frequent, but ...
- the domain is contiguous (no “holes”): $\mathbb{Z}$ is “discrete”.

Floating point types:

- Overflow and Underflow seldom, but ...
- there are holes: $\mathbb{R}$ is “continuous”.

Holes in the Domain

```
public class Main {
    public static void main(String[] args) {
    Out.print("First number =? ");
    float n1 = In.readFloat();

    Out.print("Second number =? ");
    float n2 = In.readFloat();

    Out.print("Their difference =? ");
    float d = In.readFloat();

    Out.print("computed difference - input difference = ");
    Out.println(n1-n2-d);
    }
}
```

input 1.1

input 1.0

input 0.1

output 2.2351742E-8

What is going on here?

Floating Point Rules — Rule 1

Rule 1

Do not test rounded floating point numbers for equality.

```
for (float i = 0.1; i != 1.0; i += 0.1){ // doesn't work!!!
    Out.println(i);
}
```

endless loop because i never becomes exactly 1

More explanations next time!

Floating Point Rules — Rule 2

Rule 2

Do not add two numbers providing very different orders of magnitude!

Assume we compute with 4 digits precision

$$\begin{aligned} & 1.000 \cdot 10^5 \\ + & 1.000 \cdot 10^0 \\ = & \ 1.00001 \cdot 10^5 \\ \text{“=”} & \ 1.000 \cdot 10^5 \quad \text{(Rounding on 4 places)} \end{aligned}$$

Addition of 1 does not provide any effect!

Floating Point Guidelines Rule 3

Rule 4

Do not subtract two numbers with a very similar value.

Cancellation problems (without further explanations).

5. Logical Values

Boolean Functions; the Type **`boolean`**; logical and relational operators; shortcut evaluation

Our Goal

```
int a = In.readInt();
if (a % 2 == 0) {
    Out.print("even");
} else {
    Out.print("odd");
}
```

Behavior depends on the value of a **boolean expression**

Boolean Values in Mathematics

Boolean expressions can take on one of two values:

true or **false**

The Type `boolean` in Java

- represents **logical values**
- Literals `true` and `false`
- Domain {***true***, ***false***}

```
boolean b = true; //Variable b with value true
```


Relational Operators

`a < b` (smaller than)
`a >= b` (greater than)
`a == b` (equals)
`a != b` (not equal)

arithmetic type $\times$ arithmetic type $\rightarrow$ `boolean`

Table of Relational Operators

	Symbol	Arity	Precedence	Associativity
smaller	<	2	11	left
greater	>	2	11	left
smaller equal	<=	2	11	left
greater equal	>=	2	11	left
equal	==	2	10	left
unequal	!=	2	10	left

arithmetic type $\times$ arithmetic type $\rightarrow$ `boolean`

Boolean Functions in Mathematics

- Boolean function

$$f : \{0,1\}^2 \rightarrow \{0,1\}$$

- 0 corresponds to “false”.
- 1 corresponds to “true”.

$\mathrm{AND}(x, y)$

$x \wedge y$

- "logical And"

$$f : \{0,1\}^2 \to \{0,1\}$$

- 0 corresponds to "false".
- 1 corresponds to "true".

x	y	$\mathrm{AND}(x,y)$
0	0	0
0	1	0
1	0	0
1	1	1

Logical Operator `&&`

`a && b` (logical and)

$\texttt{boolean} \times \texttt{boolean} \to \texttt{boolean}$

```
int n = -1;
int p = 3;
boolean b = (n < 0) && (0 < p); // b = true
```


$\mathrm{OR}(x, y)$

$x \vee y$

- "logical Or"

$$f : \{0,1\}^2 \to \{0,1\}$$

- 0 corresponds to "false".
- 1 corresponds to "true".

x	y	$\mathrm{OR}(x,y)$
0	0	0
0	1	1
1	0	1
1	1	1

Logical Operator `||`

`a || b` (logical or)

$\texttt{boolean} \times \texttt{boolean} \to \texttt{boolean}$

```
int n = 1;
int p = 0;
boolean b = (n < 0) || (0 < p); // b = false
```

$\mathrm{NOT}(x)$ $\neg x$

- "logical Not"

$$f : \{0, 1\} \to \{0, 1\}$$

- 0 corresponds to "false".
- 1corresponds to "true".

x	$\mathrm{NOT}(x)$
0	1
1	0

Logical Operator `!`

`!b` (logical not)

`boolean` → `boolean`

```
int n = 1;
boolean b = !(n < 0); // b = true
```


Precedences

```
!b && a
   ⇳
(!b) && a

a && b || c && d
      ⇳
(a && b) || (c && d)

a || b   &&   c || d
      ⇳
a || (b && c) || d
```


Table of Logical Operators

	Symbol	Arity	Precedence	Associativity
Logical and (AND)	`&&`	2	6	left
Logical or (OR)	`||`	2	5	left
Logical not (NOT)	`!`	1	16	right

Precedences

The unary logical operator `!`
binds more strongly than
binary arithmetic operators. These
bind more strongly than
relational operators,
and these bind more strongly than
binary logical operators.

```
7 + x < y && y != 3 * z || ! b
7 + x < y && y != 3 * z || (!b)
```


DeMorgan Rules

- `!(a && b) == (!a || !b)`
- `!(a || b) == (!a && !b)`

`!` (rich *and* beautiful) == (poor *or* ugly)

Short circuit Evaluation

- Logical operators `&&` and `||` evaluate the *left operand first*.
- If the result is then known, the right operand will *not be* evaluated.

```
x != 0 && z / x > y
```

$\Rightarrow$ No division by 0