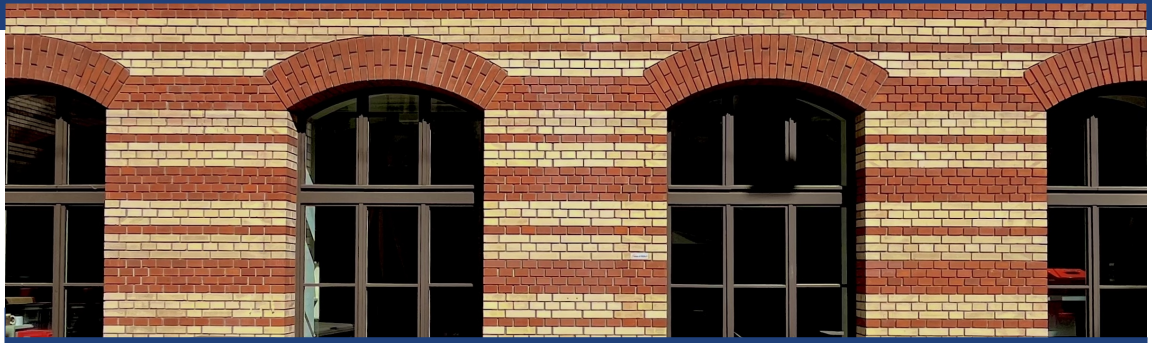


Exercise Session — Computer Science — 04

Binary representation of floating point numbers, Normalized floating point number system, Guidelines for floating point numbers, Floating point number comparisons

Overview

Today's Plan

Repetition
Binary Representation
Normalized Floating Point Systems
Floating Point Guidelines
Comparing Floating Point Numbers
Old Exam Question

1. Feedback regarding **code** expert

General things regarding **code** expert

Any questions regarding **code** expert on your part?

2. Repetition

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

`5 < 4 < 1`

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

```
5 < 4 < 1
(5 < 4) < 1
```

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

```
5 < 4 < 1
(5 < 4) < 1
false < 1
```

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

```
5 < 4 < 1
(5 < 4) < 1
false < 1
0 < 1
```

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

```
5 < 4 < 1
(5 < 4) < 1
false < 1
0 < 1
true
```

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

```
5 < 4 < 1
(5 < 4) < 1
false < 1
0 < 1
true
```

Solution 2

```
true > false
```

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

```
5 < 4 < 1
(5 < 4) < 1
false < 1
0 < 1
true
```

Solution 2

```
true > false
1 > 0
```

Expressions

Exercise

Evaluate the following expressions:

1. `5 < 4 < 1`
2. `true > false`

Solution 1

```
5 < 4 < 1
(5 < 4) < 1
false < 1
0 < 1
true
```

Solution 2

```
true > false
1 > 0
true
```

Binary Representation ...but which one?

Math. Dec	Math. Bin	`int`	`unsigned int`

Binary Representation ...but which one?

Math. Dec	Math. Bin	`int`	`unsigned int`
42_{10}			

Binary Representation ...but which one?

Math. Dec	Math. Bin	`int`	`unsigned int`
42_{10}	101010_2	`0101010`	`101010`

Binary Representation ...but which one?

Math. Dec	Math. Bin	`int`	`unsigned int`
42_{10}	101010_2	`0101010`	`101010`
-42_{10}			

Binary Representation...but which one?

Math. Dec	Math. Bin	`int`	`unsigned int`
42_{10}	101010_2	0101010	101010
-42_{10}	-101010_2	1010110	-

Binary Representation...but which one?

Math. Dec	Math. Bin	`int`	`unsigned int`
42_{10}	101010_2	0101010	101010
-42_{10}	-101010_2	1010110	-

`int` saves numbers in *two's complement representation*, thus the leading 0.

Binary Arithmetic

Tasks

1. Convert the whole numbers $a = 4$ and $b = 7$ into their binary representations (not two's complement)
2. Add the two (in their binary representation)
3. Convert the result back into decimal representation

Binary Arithmetic

Tasks

1. Convert the whole numbers $a = 4$ and $b = 7$ into their binary representations (not two's complement)
2. Add the two (in their binary representation)
3. Convert the result back into decimal representation

Solution

$a = 4_{10} = 100_2$

Binary Arithmetic

Tasks

1. Convert the whole numbers $a = 4$ and $b = 7$ into their binary representations (not two's complement)
2. Add the two (in their binary representation)
3. Convert the result back into decimal representation

Solution

$a = 4_{10} = 100_2$
$b = 7_{10} = 111_2$

Binary Arithmetic

Tasks

1. Convert the whole numbers $a = 4$ and $b = 7$ into their binary representations (not two's complement)
2. Add the two (in their binary representation)
3. Convert the result back into decimal representation

Solution

$a = 4_{10} = 100_2$
$b = 7_{10} = 111_2$
$100_2 + 111_2 = 1011_2$

Binary Arithmetic

Tasks

1. Convert the whole numbers $a = 4$ and $b = 7$ into their binary representations (not two's complement)
2. Add the two (in their binary representation)
3. Convert the result back into decimal representation

Solution

$a = 4_{10} = 100_2$
$b = 7_{10} = 111_2$
$100_2 + 111_2 = 1011_2$
$1011_2 = 11_{10}$

Questions?

3. Binary Representation

Binary Representation

binary	1	1	1	1	.	1	1	1
decimal	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

Binary Representation

binary	1	1	1	1	.	1	1	1
decimal	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

Example

$$101.011_2$$

Binary Representation

binary	1	1	1	1	.	1	1	1
decimal	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

Example

$$101.011_2 = 2^2 + 2^0 + 2^{-2} + 2^{-3}$$

Binary Representation

binary	1	1	1	1	.	1	1	1
decimal	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

Example

$$101.011_2 = 2^2 + 2^0 + 2^{-2} + 2^{-3} = 4 + 1 + \frac{1}{4} + \frac{1}{8}$$

Binary Representation

binary	1	1	1	1	.	1	1	1
decimal	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}	2^{-3}
	8	4	2	1	.	$\frac{1}{2}$	$\frac{1}{4}$	$\frac{1}{8}$

Example

$$101.011_2 = 2^2 + 2^0 + 2^{-2} + 2^{-3} = 4 + 1 + \frac{1}{4} + \frac{1}{8} = 5.375_{10}$$

Conversion to Binary Representation

Conversion to Binary Representation

Procedure (from the Lecture)

Conversion to Binary Representation

Procedure (from the Lecture)

1. Write down the number
2. Copy first digit of nunmber
3. Calculate: number − first digit
4. Multiply by 2 and start a new line

Conversion to Binary Representation

Procedure (from the Lecture)

1. Write down the number
2. Copy first digit of nunmber
3. Calculate: number $-$ first digit
4. Multiply by 2 and start a new line

Example

$x = 1.9$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
1.9			

Conversion to Binary Representation

Procedure (from the Lecture)

1. Write down the number
2. Copy first digit of nunmber
3. Calculate: number $-$ first digit
4. Multiply by 2 and start a new line

Example

$x = 1.9$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
1.9	1	0.9	1.8
1.8	1	0.8	1.6

Conversion to Binary Representation

Procedure (from the Lecture)

1. Write down the number
2. Copy first digit of nunmber
3. Calculate: number $-$ first digit
4. Multiply by 2 and start a new line

Example

$x = 1.9$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
1.9	1	0.9	1.8
1.8	1	0.8	1.6
$\mathbf{1.6}$	1	0.6	1.2

Conversion to Binary Representation

Procedure (from the Lecture)

1. Write down the number
2. Copy first digit of nunmber
3. Calculate: number $-$ first digit
4. Multiply by 2 and start a new line

Example

$x = 1.9$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
1.9	1	0.9	1.8
1.8	1	0.8	1.6
$\mathbf{1.6}$	1	0.6	1.2
1.2	1	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
$\mathbf{1.6}$	1	0.6	1.2

Conversion to Binary Representation

Procedure (from the Lecture)

1. Write down the number
2. Copy first digit of nunmber
3. Calculate: number $-$ first digit
4. Multiply by 2 and start a new line

Example

$x = 1.9$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
1.9	1	0.9	1.8
1.8	1	0.8	1.6
$\mathbf{1.6}$	1	0.6	1.2
1.2	1	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
$\mathbf{1.6}$	1	0.6	1.2

Notice the $\mathbf{1.6}$ above?

$1.6_{10} = 1.1\overline{1100}_2$

Conversion to Binary Representation

Procedure (from the Lecture)

1. Write down the number
2. Copy first digit of nunmber
3. Calculate: number $-$ first digit
4. Multiply by 2 and start a new line

Example

$x = 1.9$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
1.9	1	0.9	1.8
1.8	1	0.8	1.6
$\mathbf{1.6}$	1	0.6	1.2
1.2	1	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
$\mathbf{1.6}$	1	0.6	1.2

Notice the $\mathbf{1.6}$ above?

$1.6_{10} = 1.1\overline{1100}_2 =$

$1.1100110011001100110011001100110011$

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

0.25_{10}

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

$0.25_{10} = 0.01_2$

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2

$0.25_{10} = 0.01_2$

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4

$0.25_{10} = 0.01_2$

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4
0.4	0	0.4	0.8

$0.25_{10} = 0.01_2$

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

$0.25_{10} = 0.01_2$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

$0.25_{10} = 0.01_2$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
1.6	1	0.6	1.2

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

$0.25_{10} = 0.01_2$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
1.6	1	0.6	1.2
1.2	1	0.2	0.4

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

$0.25_{10} = 0.01_2$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
1.6	1	0.6	1.2
1.2	1	0.2	0.4
0.4	0	0.4	0.8

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

$0.25_{10} = 0.01_2$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
1.6	1	0.6	1.2
1.2	1	0.2	0.4
0.4	0	0.4	0.8

11.1_{10}

Decimal to Binary

Task Compute the binary expansions of 0.25_{10} and 11.1_{10}.
Solution

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.25	0	0.25	0.5
0.5	0	0.5	1
1	1	0	0

$0.25_{10} = 0.01_2$

x	b_i	$x - b_i$	$2 \cdot (x - b_i)$
0.1	0	0.1	0.2
0.2	0	0.2	0.4
0.4	0	0.4	0.8
0.8	0	0.8	1.6
1.6	1	0.6	1.2
1.2	1	0.2	0.4
0.4	0	0.4	0.8

$11.1_{10} = 1011.0\overline{0011}_2$

Questions?

4. Normalized Floating Point Systems

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

* Nomalized ($d_0 \neq 0$)

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)
e_{min}	Smallest possible exponent

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)
e_{min}	Smallest possible exponent
e_{max}	Largest possible exponent

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$ …describes numbers of the form:

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)
e_{min}	Smallest possible exponent
e_{max}	Largest possible exponent

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)
e_{min}	Smallest possible exponent
e_{max}	Largest possible exponent

…describes numbers of the form:

$$\pm d_0.d_1d_2d_3 \ldots d_{p-1} \cdot \beta^e$$

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)
e_{min}	Smallest possible exponent
e_{max}	Largest possible exponent

…describes numbers of the form:

$$\pm d_0.d_1d_2d_3 \ldots d_{p-1} \cdot \beta^e$$

where:
$d_i \in \{0, \ldots, b-1\}$

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)
e_{min}	Smallest possible exponent
e_{max}	Largest possible exponent

…describes numbers of the form:

$$\pm d_0.d_1d_2d_3 \ldots d_{p-1} \cdot \beta^e$$

where:
$d_i \in \{0, \ldots, b-1\}$
$d_0 \neq 0$

Normalized Floating Point Systems

$F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$

*	Nomalized ($d_0 \neq 0$)
$\beta \geq 2$	Basis
$p \geq 1$	Precision (Number of Digits)
e_{min}	Smallest possible exponent
e_{max}	Largest possible exponent

…describes numbers of the form:

$$\pm d_0.d_1d_2d_3 \ldots d_{p-1} \cdot \beta^e$$

where:
$d_i \in \{0, \ldots, b-1\}$
$d_0 \neq 0$
$e \in [e_{\text{min}}, e_{\text{max}}]$

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$$
\begin{aligned}
0.000_2 \cdot 2^1 &= 0_{10} \\
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.0001_2 \cdot 2^{-1} &= 0.53125_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10} \\
1.111_2 \cdot 2^5 &= 60_{10}
\end{aligned}
$$

Exercises

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$$
\begin{aligned}
0.000_2 \cdot 2^1 &= 0_{10} \\
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.0001_2 \cdot 2^{-1} &= 0.53125_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10} \\
1.111_2 \cdot 2^5 &= 60_{10}
\end{aligned}
$$

Solutions

in F^*

Exercises

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$$
\begin{aligned}
0.000_2 \cdot 2^1 &= 0_{10} \\
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.0001_2 \cdot 2^{-1} &= 0.53125_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10} \\
1.111_2 \cdot 2^5 &= 60_{10}
\end{aligned}
$$

Solutions

in F^*

$$
\begin{aligned}
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10}
\end{aligned}
$$

Exercises

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$$
\begin{aligned}
0.000_2 \cdot 2^1 &= 0_{10} \\
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.0001_2 \cdot 2^{-1} &= 0.53125_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10} \\
1.111_2 \cdot 2^5 &= 60_{10}
\end{aligned}
$$

Solutions

in F^*

$$
\begin{aligned}
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10}
\end{aligned}
$$

not in F^*

Exercises

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$$
\begin{aligned}
0.000_2 \cdot 2^1 &= 0_{10} \\
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.0001_2 \cdot 2^{-1} &= 0.53125_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10} \\
1.111_2 \cdot 2^5 &= 60_{10}
\end{aligned}
$$

Solutions

in F^*

$$
\begin{aligned}
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10}
\end{aligned}
$$

not in F^*

$$0.000_2 \cdot 2^1$$

Exercises

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$0.000_2 \cdot 2^1 = 0_{10}$
$1.000_2 \cdot 2^1 = 2_{10}$
$1.001_2 \cdot 2^{-1} = 0.5625_{10}$
$1.0001_2 \cdot 2^{-1} = 0.53125_{10}$
$1.111_2 \cdot 2^{-2} = 0.46875_{10}$
$1.111_2 \cdot 2^5 = 60_{10}$

Solutions

in F^*

$1.000_2 \cdot 2^1 = 2_{10}$
$1.001_2 \cdot 2^{-1} = 0.5625_{10}$
$1.111_2 \cdot 2^{-2} = 0.46875_{10}$

not in F^*

$0.000_2 \cdot 2^1$ not "normalizable"
$1.0001_2 \cdot 2^{-1}$

Exercises

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$0.000_2 \cdot 2^1 = 0_{10}$
$1.000_2 \cdot 2^1 = 2_{10}$
$1.001_2 \cdot 2^{-1} = 0.5625_{10}$
$1.0001_2 \cdot 2^{-1} = 0.53125_{10}$
$1.111_2 \cdot 2^{-2} = 0.46875_{10}$
$1.111_2 \cdot 2^5 = 60_{10}$

Solutions

in F^*

$1.000_2 \cdot 2^1 = 2_{10}$
$1.001_2 \cdot 2^{-1} = 0.5625_{10}$
$1.111_2 \cdot 2^{-2} = 0.46875_{10}$

not in F^*

$0.000_2 \cdot 2^1$ not "normalizable"
$1.0001_2 \cdot 2^{-1}$ $5 > p = 4$
$1.111_2 \cdot 2^5$

Exercises

Exercises

Are the following numbers in the set $F^*(2, 4, -2, 2)$?

$$
\begin{aligned}
0.000_2 \cdot 2^1 &= 0_{10} \\
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.0001_2 \cdot 2^{-1} &= 0.53125_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10} \\
1.111_2 \cdot 2^5 &= 60_{10}
\end{aligned}
$$

Solutions

in F^*

$$
\begin{aligned}
1.000_2 \cdot 2^1 &= 2_{10} \\
1.001_2 \cdot 2^{-1} &= 0.5625_{10} \\
1.111_2 \cdot 2^{-2} &= 0.46875_{10}
\end{aligned}
$$

not in F^*

$$
\begin{aligned}
0.000_2 \cdot 2^1 &\quad \text{not "normalizable"} \\
1.0001_2 \cdot 2^{-1} &\quad 5 > p = 4 \\
1.111_2 \cdot 2^5 &\quad 5 \notin [-2, 2]
\end{aligned}
$$

more exercises

Exercise

Name the following numbers in $F^*(2, 4, -2, 2)$ in decimal representation

1. the largest number
2. the smallest number
3. the smallest non-negative number

more exercises

Exercise

Name the following numbers in $F^*(2, 4, -2, 2)$ in decimal representation

1. the largest number
2. the smallest number
3. the smallest non-negative number

Solutions

largest:

more exercises

Exercise

Name the following numbers in $F^*(2, 4, -2, 2)$ in decimal representation

1. the largest number
2. the smallest number
3. the smallest non-negative number

Solutions

largest: $1.111 \cdot 2^2 = 7.5_{10}$
smallest:

more exercises

Exercise

Name the following numbers in $F^*(2, 4, -2, 2)$ in decimal representation

1. the largest number
2. the smallest number
3. the smallest non-negative number

Solutions

largest: $1.111 \cdot 2^2 = 7.5_{10}$
smallest: $-1.111 \cdot 2^2 = -7.5_{10}$
smallest > 0:

more exercises

Exercise

Name the following numbers in $F^*(2, 4, -2, 2)$ in decimal representation

1. the largest number
2. the smallest number
3. the smallest non-negative number

Solutions

largest:	$1.111 \cdot 2^2 = 7.5_{10}$
smallest:	$-1.111 \cdot 2^2 = -7.5_{10}$
smallest > 0:	$1.000 \cdot 2^{-2} = 0.25_{10}$

more exercises

Exercise

Name the following numbers in $F^*(2, 4, -2, 2)$ in decimal representation

1. the largest number
2. the smallest number
3. the smallest non-negative number

Solutions

largest: $1.111 \cdot 2^2 = 7.5_{10}$
smallest: $-1.111 \cdot 2^2 = -7.5_{10}$
smallest > 0: $1.000 \cdot 2^{-2} = 0.25_{10}$

Trick

For a given $F^*(\beta, p, e_{\text{min}}, e_{\text{max}})$:

Largest: $1.11\ldots1 \cdot 2^{e_{\text{max}}}$
Smallest: $-$Largest
Smallest > 0: $1.00\ldots0 \cdot 2^{e_{\text{min}}}$

Normalized Floating Point Systems

Question
How many numbers are in $F^*(2, 4, -2, 2)$?

Normalized Floating Point Systems

Question
How many numbers are in $F^*(2, 4, -2, 2)$?

Answer
For a fixed exponent there are always three digits that can be varied freely and for all numbers there is the negative one in the set F^*. This results in $2 \cdot 2^3 = 16$ numbers per exponent. There are 5 possible exponents, therefore $5 \cdot 16 = 80$ numbers. Note that no number is counted twice, as each NFP is *unique*.

Questions?

Arithmetic in F^*

Adding Floats

1. Reformulate both in terms of the same exponent
2. Add them in their binary representation
3. Renormalize the sum
4. Round if necessary

Example

$$F^*(2, 6, -2, 3)$$
$$1.125_{10} + 9.25_{10}$$

Example

$F^*(2, 6, -2, 3)$

$1.125_{10} + 9.25_{10}$

$1.001_2 + 1001.01_2$ (already in terms of same exponent)

Example

$$
\begin{aligned}
F^*(2, 6, -2, 3) & \\
1.125_{10} + 9.25_{10} & \\
1.001_2 + 1001.01_2 & \quad \text{(already in terms of same exponent)} \\
1010.011 & \quad \text{Addition}
\end{aligned}
$$

Example

$$
\begin{aligned}
F^*(2, 6, -2, 3) & \\
1.125_{10} + 9.25_{10} & \\
1.001_2 + 1001.01_2 & \quad \text{(already in terms of same exponent)} \\
1010.011 & \quad \text{Addition} \\
1.010011 \cdot 2^3 & \quad \text{Renormalizing, adapt } e \text{ and } p
\end{aligned}
$$

Example

$F^*(2, 6, -2, 3)$	
$1.125_{10} + 9.25_{10}$	
$1.001_2 + 1001.01_2$	(already in terms of same exponent)
1010.011	Addition
$1.010011 \cdot 2^3$	Renormalizing, adapt e and p
$1.01010 \cdot 2^3$	Round: 1 up (and potentially carry), 0 down

Example

$$
\begin{aligned}
F^*(2, 6, -2, 3) & \\
1.125_{10} + 9.25_{10} & \\
1.001_2 + 1001.01_2 & \quad \text{(already in terms of same exponent)} \\
1010.011 & \quad \text{Addition} \\
1.010011 \cdot 2^3 & \quad \text{Renormalizing, adapt } e \text{ and } p \\
1.01010 \cdot 2^3 & \quad \text{Round: } 1 \text{ up (and potentially carry), } 0 \text{ down} \\
1.01010_2 \cdot 2_2^3 = 1010.10_2 = 10.5_{10} &
\end{aligned}
$$

Example

$$
\begin{aligned}
F^*(2, 6, -2, 3) & \\
1.125_{10} + 9.25_{10} & \\
1.001_2 + 1001.01_2 & \quad \text{(already in terms of same exponent)} \\
1010.011 & \quad \text{Addition} \\
1.010011 \cdot 2^3 & \quad \text{Renormalizing, adapt } e \text{ and } p \\
1.01010 \cdot 2^3 & \quad \text{Round: } 1 \text{ up (and potentially carry), } 0 \text{ down} \\
1.01010_2 \cdot 2_2^3 = 1010.10_2 = 10.5_{10} & \quad \neq 10.375_{10}
\end{aligned}
$$

Why 10.5 and not 10.375?

Why 10.5 and not 10.375?

Simply because the precise number 10.375 is not representable in the given F^* and 10.5 is the closest representable one in F^*. This is the reason floats can be dangerous to handle. This is why we have follow the *Floating Point Guidelines*.

It is not 10.25, because in this case we round up, although the difference between 10.25 and 10.5 and 10.375 is 0.125.

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Solution

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Solution

1. Bring both to the same exponent, say -1

$$1.001 \cdot 2^{-1} + 0.1111 \cdot 2^{-1}$$

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Solution

1. Bring both to the same exponent, say -1

$$1.001 \cdot 2^{-1} + 0.1111 \cdot 2^{-1}$$

2. Add them in binary notation:

$$10.0001 \cdot 2^{-1}$$

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Solution

1. Bring both to the same exponent, say -1

$$1.001 \cdot 2^{-1} + 0.1111 \cdot 2^{-1}$$

2. Add them in binary notation:

$$10.0001 \cdot 2^{-1}$$

3. Renormalize:

$$1.00001 \cdot 2^{0}$$

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Solution

1. Bring both to the same exponent, say -1

$$1.001 \cdot 2^{-1} + 0.1111 \cdot 2^{-1}$$

2. Add them in binary notation:

$$10.0001 \cdot 2^{-1}$$

3. Renormalize:

$$1.00001 \cdot 2^{0}$$

4. Round: $1.000 \cdot 2^0$

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Solution

1. Bring both to the same exponent, say -1

$$1.001 \cdot 2^{-1} + 0.1111 \cdot 2^{-1}$$

2. Add them in binary notation:

$$10.0001 \cdot 2^{-1}$$

3. Renormalize:

$$1.00001 \cdot 2^{0}$$

4. Round: $1.000 \cdot 2^0 = 1_{10}$

Exercise

Exercise

Add

$$1.001 \cdot 2^{-1} = 0.5625_{10}$$

to

$$1.111 \cdot 2^{-2} = 0.46875_{10}$$

in

$$F^*(2, 4, -2, 2).$$

Solution

1. Bring both to the same exponent, say -1

$$1.001 \cdot 2^{-1} + 0.1111 \cdot 2^{-1}$$

2. Add them in binary notation:

$$10.0001 \cdot 2^{-1}$$

3. Renormalize:

$$1.00001 \cdot 2^{0}$$

4. Round: $1.000 \cdot 2^0 = 1_{10} \neq 1.03125_{10}$

Questions?

5. Floating Point Guidelines

Guidelines

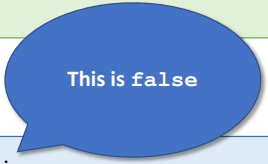
Guideline 1:

> «Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»

Guideline 1 – Example

Guideline 1:

«Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»



Example:

```
float a = 1.05f;
if (100*a == 105.0f)
    std::cout << "no output\n";
```

Guideline 1 – Example

Guideline 1:

«Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»

Example:

```
float a = 1.05f;
if (100*a == 105.0f)
    std::cout << "no output\n";
```

This is `false`

Problem:

`1.05f` not representable

Guideline 1 – Example

Guideline 1:

«Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»

This is `false`

Example:

```
float a = 1.05f;
if (100*a == 105.0f)
    std::cout << "no output\n";
```

Problem:

`1.05f` not representable

$$1.05 = \overbrace{1.00001100110011001100110}^{\text{24bit}}\ldots \cdot 2^0$$

(rounding) $\rightarrow 1.049999995231\ldots = 1.00001100110011001100110 \cdot 2^0$

Guidelines

Guideline 1:

«Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»

Guideline 2:

«**Avoid** the **addition** of numbers of extremely **different sizes**!»

Guideline 2 – Example

Guideline 2:

«**Avoid** the **addition** of numbers of extremely **different sizes**!»

Example:

```
float a = 67108864.0f + 1.0f;

if (a > 67108864.0f)
    std::cout << "This is not output ... \n";
```

Guideline 2 – Example

Guideline 2:

«**Avoid** the **addition** of numbers of extremely **different sizes**!»

Problem:

Significand too short

Example:

```
float a = 67108864.0f + 1.0f;

if (a > 67108864.0f)
    std::cout << "This is not output ... \n";
```

Guideline 2 – Example

Guideline 2:

«**Avoid** the **addition** of numbers of extremely **different sizes**!»

Problem:

Significand too short

Example:

```
float a = 67108864.0f + 1.0f;

if (a > 67108864.0f)
    std::cout << "This is not output ... \n";
```

$$
\begin{aligned}
67108864 &= \overbrace{1.00000000000000000000000}^{\text{24bit}} \cdot 2^{26} \\
+1 &= 0.00000000000000000000000001 \cdot 2^{26} \\
\hline
67108865 &= 1.00000000000000000000000001 \cdot 2^{26}
\end{aligned}
$$

Guideline 2 – Example

Guideline 2:

«**Avoid** the **addition** of numbers of extremely **different sizes**!»

Problem:

Significand too short

Example:

```
float a = 67108864.0f + 1.0f;

if (a > 67108864.0f)
    std::cout << "This is not output ... \n";
```

24bit

$$
\begin{aligned}
67108864 &= \overbrace{1.00000000000000000000000}\cdot 2^{26} \\
+1 &= 0.00000000000000000000000001 \cdot 2^{26} \\
\hline
67108865 &= 1.00000000000000000000000001 \cdot 2^{26} \\
\text{(rounding)} \rightarrow 67108864 &= 1.00000000000000000000000 \cdot 2^{26}
\end{aligned}
$$

Guidelines

Guideline 1:

«Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»

Guideline 2:

«**Avoid** the **addition** of numbers of extremely **different sizes**!»

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- Consider sequence $\mathrm{x}_{n+1} = 6x_n - 1$

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- Consider sequence $x_{n+1} = 6x_n - 1$
- Computing some sequences for given x_0:

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- Consider sequence $x_{n+1} = 6x_n - 1$
- Computing some sequences for given x_0:
  - e.g. $x_0 = 1$ → $x_1 = 5, \quad x_2 = 29, \quad x_3 = 173, \quad \ldots$

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- Consider sequence $x_{n+1} = 6x_n - 1$
- Computing some sequences for given x_0:
  - e.g. $x_0 = 1$ → $x_1 = 5, \quad x_2 = 29, \quad x_3 = 173, \quad \ldots$
  - e.g. $x_0 = 0.2$ → $x_1 = 0.2, \quad x_2 = 0.2, \quad x_3 = 0.2, \quad \ldots$

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- Consider sequence $x_{n+1} = 6x_n - 1$
- Computing some sequences for given x_0:
  - e.g. $x_0 = 1$ → $x_1 = 5, \quad x_2 = 29, \quad x_3 = 173, \quad \ldots$
  - e.g. $x_0 = 0.2$ → $x_1 = 0.2, \quad x_2 = 0.2, \quad x_3 = 0.2, \quad \ldots$

C++ claims

$x_{14} \approx 622.982$

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- What went wrong?

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- What went wrong?
  - `float` represents 0.2 as 0.20000000298…
  - Thus: $6 \cdot x_0 - 1 \neq 1.2 - 1$

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- What went wrong?
  - `float` represents 0.2 as 0.20000000298…
  - Thus: $6 \cdot x_0 - 1 \neq 1.2 - 1$ but rather:

$$x_1 = 0.20000004768 \ldots$$
$$x_2 = 0.20000028610 \ldots$$
$$x_3 = 0.20000171661 \ldots$$
$$\vdots$$

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

Example:

- What went wrong?
  - `float` represents 0.2 as 0.20000000298…
  - Thus: $6 \cdot x_0 - 1 \neq 1.2 - 1$ but rather:

$$x_1 = 0.2000000\mathbf{4768} \ldots$$
$$x_2 = 0.2000000\mathbf{28610} \ldots$$
$$x_3 = 0.2000\mathbf{0171661} \ldots$$
$$\vdots$$

Note how error increases!

Guideline 3 – Example

Guideline 3:

«**Avoid** the **subtraction** of numbers of **similar sizes**!»

But why do we subtract two similarly sized floating point numbers when we are comparing them to see if they are (roughly) equal?

In these cases, we do not further use the result from the subtraction in any other calculations. Therefore, the error does not add up over time causing issues as seen in the previous example.

6. Comparing Floating Point Numbers

Comparing Floating Point Numbers

In short: do not test for *equality*
rather check for "*within tolerance*"

```
// Example of "equality" check function for doubles
bool equal(double x, double y, double tol){
    double diff = x - y;
    if(diff < 0){
        diff *= -1;
    }
    return (diff < tol);
}
```

The Comparison Problem

- Given `fp1` and `fp2` of type `float` or `double`.

- Guideline 1:

  «Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»

The Comparison Problem

- Given `fp1` and `fp2` of type `float` or `double`.
- Guideline 1:

  «Do **not** test two floating point numbers for **equality**, if at least one of them was rounded before.»

- Thus `fp1 == fp2` should be **avoided**.

The Comparison Problem

- How can we compare instead?

The Comparison Problem

- How can we compare instead?

- First idea:
  Allow for small differences!

> Given: tolerance value `c > 0`.
>
> **`fp1`** **"equals"** **`fp2`** whenever **`|fp1 - fp2| < c`**

(Remark: |…| means absolute value. In C++ it's not available using vertical bars.)

The Comparison Problem

Given: tolerance value `c > 0`.

`fp1` "equals" `fp2` whenever **`|fp1 - fp2| < c`**

- Examples (`c` is `0.001`):
  - `fp1 = 10.0` and `fp2 = 12.0`

(Remark: on this slide = is meant in the mathematical sense.)

The Comparison Problem

Given: tolerance value `c > 0`.

`fp1` "equals" `fp2` whenever **`|fp1 - fp2| < c`**

- Examples (`c` is `0.001`):
  - `fp1 = 10.0` and `fp2 = 12.0`
    `|10.0 - 12.0| = 2.0`

(Remark: on this slide = is meant in the mathematical sense.)

The Comparison Problem

Given: tolerance value `c > 0`.

`fp1` "equals" `fp2` whenever **`|fp1 – fp2| < c`**

- Examples (`c` is `0.001`):
  - `fp1 = 10.0` and `fp2 = 12.0`
    `|10.0 – 12.0| = 2.0 > c`
    Thus: **not "equal"**

(Remark: on this slide = is meant in the mathematical sense.)

The Comparison Problem

Given: tolerance value `c > 0`.

`fp1` "equals" `fp2` whenever **`|fp1 – fp2| < c`**

- Examples (`c` is `0.001`):
  - `fp1 = 10.0` and `fp2 = 12.0`
    `|10.0 – 12.0| = 2.0 > c`
    Thus: **not "equal"**
  - `fp1 = 10.0` and `fp2 = 10.000013`

(Remark: on this slide = is meant in the mathematical sense.)

The Comparison Problem

Given: tolerance value `c > 0`.

`fp1` "equals" `fp2` whenever **`|fp1 – fp2| < c`**

- Examples (`c` is `0.001`):
  - `fp1 = 10.0` and `fp2 = 12.0`
    `|10.0 – 12.0| = 2.0 > c`
    Thus: **not "equal"**
  - `fp1 = 10.0` and `fp2 = 10.000013`
    `|10.0 – 10.000013| = 0.000013`

(Remark: on this slide = is meant in the mathematical sense.)

The Comparison Problem

Given: tolerance value `c > 0`.

`fp1` "equals" `fp2` whenever **`|fp1 – fp2| < c`**

- Examples (`c` is `0.001`):
  - `fp1 = 10.0` and `fp2 = 12.0`
    `|10.0 – 12.0| = 2.0 > c`
    Thus: **not "equal"**
  - `fp1 = 10.0` and `fp2 = 10.000013`
    `|10.0 – 10.000013| = 0.000013 < c`
    Thus: **"equal"**

(Remark: on this slide = is meant in the mathematical sense.)

Exercise

Write the following function:

```
// POST: returns true if and only if
//       |x - y| < tol
bool equals (double x, double y, double tol) {
  ...
}
```

Exercise

For example:

```
// POST: returns true if and only if
//       |x - y| < tol
bool equals (double x, double y, double tol) {
  double diff = x - y;
  if (diff < 0)
    diff *= -1; // absolute value
  return diff < tol;
}
```

Remark

- Comparing absolute differences with a tolerance value is a great first idea!

- (But: for example problems when the numbers are large.)

7. Old Exam Question

Exam Question

Geben Sie ein möglichst knappes normalisiertes Fliesskommazahlensystem an, mit welchem sich die folgenden dezimalen Werte gerade noch genau darstellen lassen: jede Verkleinerung von p, $e_{\max}$ oder $-e_{\min}$ muss dazu führen, dass mindestens eine Zahl nicht mehr dargestellt werden kann.
Hinweis: p zählt auch die führende Ziffer.
Tipp: Schreiben Sie sich die normalisierte Binärzahldarstellung der Werte auf, wenn sie für Sie nicht offensichtlich ist.

Provide a smallest possible normalized floating point number system that can still represent the following values exactly: any decrease of the numbers p, $e_{\max}$ or $-e_{\min}$ must imply that at least one of the numbers cannot be represented any more.
Hint: p does also count the leading digit.
Tip: Write down the normalized binary representation of the values, if it is not obvious for you.

Werte / *Values*: 2.25, $\frac{1}{8}$, 0.5, 16.5, 2^3

$F^*(\beta, p, e_{\min}, e_{\max})$ mit / *with*

$\beta = 2$, $p =$, $e_{\min} =$, $e_{\max} =$.

Exam Question

Geben Sie ein möglichst knappes normalisiertes Fliesskommazahlensystem an, mit welchem sich die folgenden dezimalen Werte gerade noch genau darstellen lassen: jede Verkleinerung von p, $e_{\max}$ oder $-e_{\min}$ muss dazu führen, dass mindestens eine Zahl nicht mehr dargestellt werden kann.
Hinweis: p zählt auch die führende Ziffer.
Tipp: Schreiben Sie sich die normalisierte Binärzahldarstellung der Werte auf, wenn sie für Sie nicht offensichtlich ist.

Provide a smallest possible normalized floating point number system that can still represent the following values exactly: any decrease of the numbers p, $e_{\max}$ or $-e_{\min}$ must imply that at least one of the numbers cannot be represented any more.
Hint: p does also count the leading digit.
Tip: Write down the normalized binary representation of the values, if it is not obvious for you.

Werte / *Values*: 2.25, $\frac{1}{8}$, 0.5, 16.5, 2^3

$F^*(\beta, p, e_{\min}, e_{\max})$ mit / *with*

$\beta = 2$, $p = 6$, $e_{\min} = -3$, $e_{\max} = 4$.

8. Outro

General Questions?

See you next time

Have a nice week!