Guidelines

Guideline 1:

«Do not test two floating point numbers for equality, if at least one of them was rounded before.»
Guideline 1:

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

Example:

```cpp
float a = 0.1f;
if (10*a == 1.0f)
    std::cout << "no output\n";
```

This is false
Guideline 1: «Do not test two floating point numbers for equality, if at least one of them was rounded before.»

Example:

```cpp
float a = 0.1f;
if (10*a == 1.0f)
    std::cout << "no output\n";
```

Problem: 0.1f not representable

This is false
Guideline 1:

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

Example:

```cpp
float a = 0.1f;
if (10*a == 1.0f)
    std::cout << "no output\n";
```

This is false

Problem:

0.1f not representable

\[
0.1 = \frac{1.100110011001100110011001}{2^4} \cdot 2^{-4} \\
\text{(rounding)} \rightarrow 0.10000000149... = \frac{1.1001100110011001100110110110011001100110110011}{2^24} \cdot 2^{-4}
\]
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:

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

Example:

```cpp
float a = 67108864.0f + 1.0f;
if (a > 67108864.0f)
    std::cout << "This is not output ... \n";
```
Guideline 2:

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

Problem:
Significand too short

Example:

```cpp
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!»

Example:

```cpp
float a = 67108864.0f + 1.0f;
if (a > 67108864.0f)
    std::cout << "This is not output ... \n";
```

Problem:
Significand too short

\[
\begin{align*}
67108864 &= 1.00000000000000000000000000000000 \cdot 2^{26} \\
+1 &= 0.00000000000000000000000000000001 \cdot 2^{26} \\
67108865 &= 1.00000000000000000000000000000001 \cdot 2^{26}
\end{align*}
\]
Guideline 2 – Example

Guideline 2:

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

Example:

```cpp
float a = 67108864.0f + 1.0f;
if (a > 67108864.0f)
    std::cout << "This is not output ... \n";
```

Problem:
Significand too short

24bit

\[
\begin{align*}
67108864 &= 1.0000000000000000000000000000000 \cdot 2^{26} \\
+1 &= 0.0000000000000000000000000000001 \cdot 2^{26} \\
67108865 &= 1.0000000000000000000000000000001 \cdot 2^{26}
\end{align*}
\]

(rounding) → 67108864 = 1.0000000000000000000000000000000 \cdot 2^{26}
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 $x_{n+1} = 6x_n - 1$
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. \( x_0 = 1 \) \( \Rightarrow \) \( x_1 = 5, \ x_2 = 29, \ x_3 = 173, \ldots \)
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. \( x_0 = 1 \) \( \rightarrow \) \( x_1 = 5, \ x_2 = 29, \ x_3 = 173, \ ... \)
  • e.g. \( x_0 = 0.2 \) \( \rightarrow \) \( x_1 = 0.2, \ x_2 = 0.2, \ x_3 = 0.2, \ ... \)
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. \( x_0 = 1 \) \( \Rightarrow \) \( x_1 = 5, \ x_2 = 29, \ x_3 = 173, \ ... \)
  • e.g. \( x_0 = 0.2 \) \( \Rightarrow \) \( x_1 = 0.2, \ x_2 = 0.2, \ x_3 = 0.2, \ ... \)

C++ claims

\( x_{14} \approx 622.982 \)
Guideline 3: «Avoid the subtraction of numbers of similar sizes!»

Example:

• What went wrong?
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:
    
    $x_1 = 0.20000004768 ...$
    
    $x_2 = 0.20000028610 ...$
    
    $x_3 = 0.20000171661 ...$
    
    $\vdots$
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:
    \[
    x_1 = 0.200000004768 \ldots \\
    x_2 = 0.20000028610 \ldots \\
    x_3 = 0.20000171661 \ldots \\
    \vdots
    \]

Note how error increases!