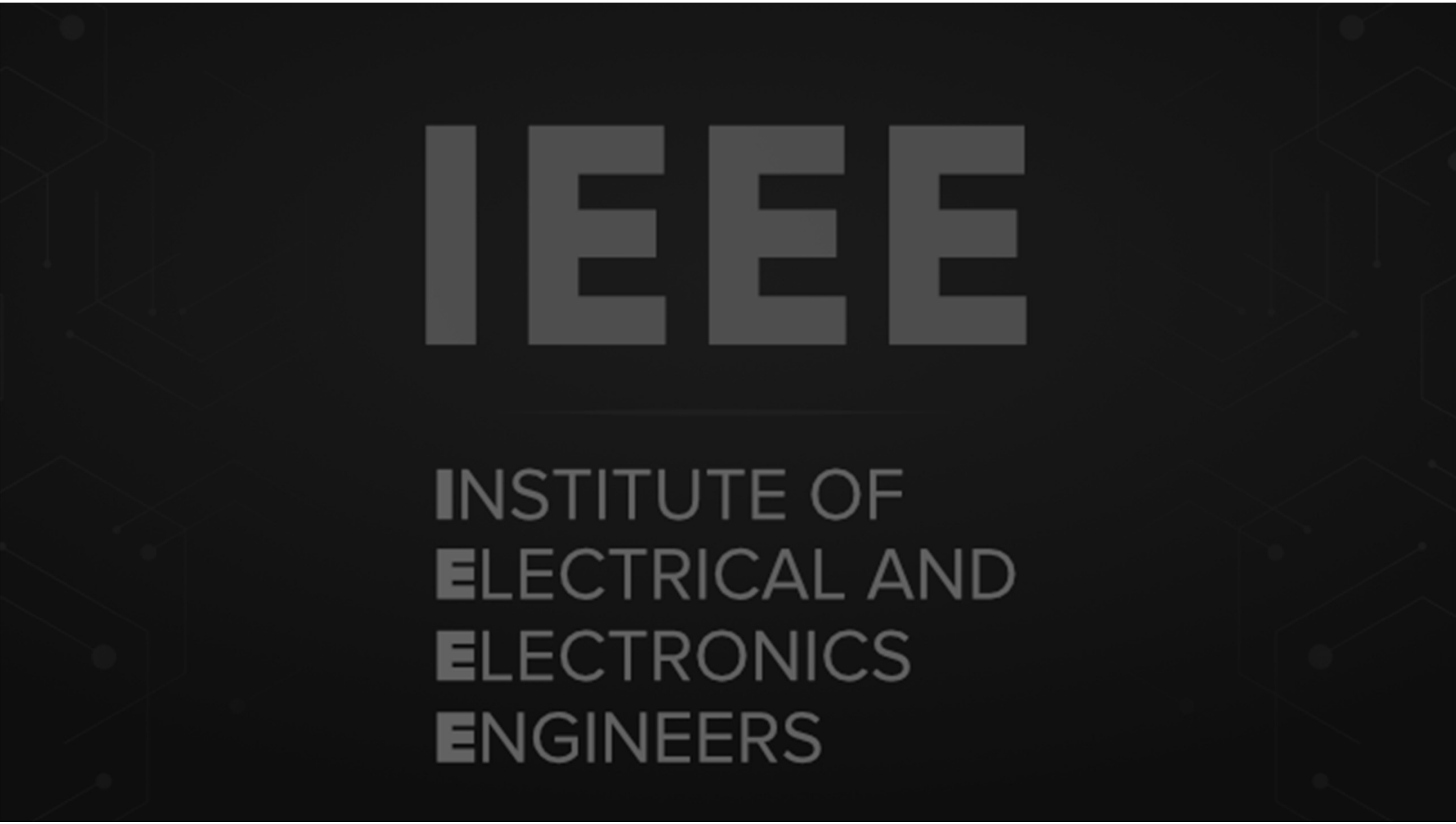
MOWNiT - Arytmetyka zmiennoprzecinkowa

Michał Bert


IEEE
INSTITUTE OF
ELECTRICAL AND
ELECTRONICS
ENGINEERS

IEEE 754

- Standard zapisu liczb zmiennoprzecinkowych
- Definiuje zapis liczb dla pojedynczej precyzji oraz podwójnej precyzji
- Określa także m.in. wartości szczególne (+0, -0, +inf, -inf, NaN) oraz reguły zaokrąglania

IEEE 754 – Zapis liczb

- Liczba składa się z trzech elementów:
  - Bit znaku – pierwszy bit liczby, ustawiany jest na 0 dla liczb dodatnich oraz 1 dla liczb ujemnych
  - Cecha – ilość bitów przeznaczona na wykładnik podstawy systemu (w naszym wypadku 2)
  - Mantysa – liczba ułamkowa
- Ilość bitów przeznaczona na cechę oraz mantysę różni się w zależności od precyzji

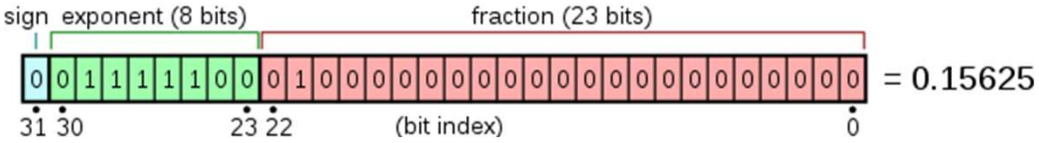
IEEE 754 – Exponent Bias

- Bias to przesunięcie wykładnika w celu reprezentacji go jako liczbę nieujemną
- Cecha liczby zapisywana jest jako ciąg bitów bez znaku, co ułatwia porównania
- W przypadku n bitów na wykładnik bias wynosi $2^{n-1} - 1$

- Np. dla typu podwójnej precyzji:
  - Ilość bitów na cechę – 8
  - Bias = $2^{8-1} - 1 = 2^7 - 1 = 128 - 1 = 127$
  - Wartość wykładnika w zapisie 8 bitów ∈ <1, 254>
  - Realna wartość wykładnika ∈ <−126, 127>

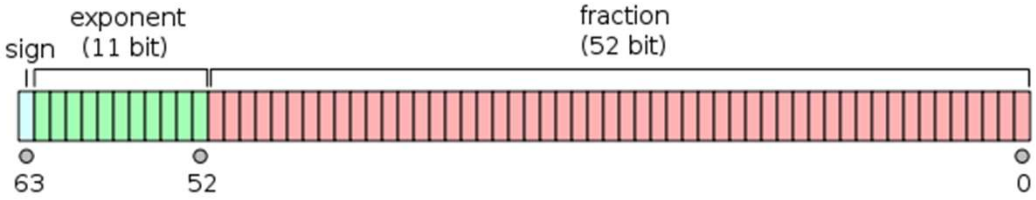
IEEE 754 - float

- Standard pojedynczej precyzji
- Cecha – 8 bitów
- Mantysa – 23 bity



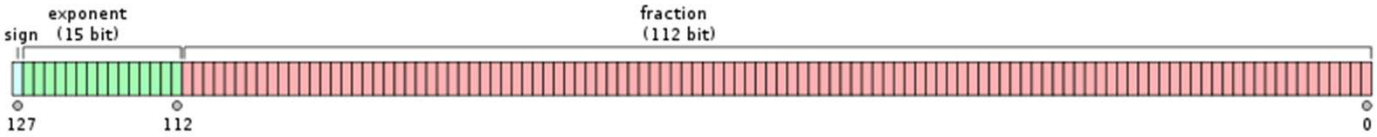
IEEE 754 - double

- Standard podwójnej precyzji
- Cecha – 11 bitów
- Mantysa – 52 bity



IEEE 754 – long double

- Standard poczwórnej precyzji
- Cecha – 15 bitów
- Mantysa – 112 bitów



IEEE 754 - 0

- +0
  - Wszystkie bity ustawione na 0
  - Otrzymane np. w wyniku dzielenia 1 przez odpowiednio wielką liczbę
- -0
  - Wszystkie bity ustawione na 0, z wyjątkiem bitu znaku
  - Otrzymane np. w wyniku dzielenia -1 przez odpowiednio wielką liczbę

IEEE 754 - inf

- +inf
  - Wszystkie bity wykładnika ustawione na 1, mantysy na 0. Bit znaku ustawiony na 0
  - Otrzymane np. w wyniku dzielenia 1.0/0.0
- -inf
  - Wszystkie bity wykładnika ustawione na 1, mantysy na 0. Bit znaku ustawiony na 1
  - Otrzymane np. w wyniku dzielenia -1.0/0.0

IEEE 754 - NaN

- NaN – Not a Number
- Wartość zwracana w przypadku niepoprawnych operacji matematycznych
- Otrzymywana np. w wyniku operacji inf * 0, sqrt(-1), itp.

Pomiary

Wartości do wyznaczenia

- Ilość bajtów użyta do przechowywania zmiennej danego typu
- Liczba bitów mantysy
- Liczba bitów cechy
- Maszynowy epsilon
- Wartości specjalne
  - +0
  - -0
  - +inf
  - -inf
  - NaN

Obliczanie rozmiaru typu danych

- Do sprawdzenia rozmiaru typu danych wykorzystany został operator ***sizeof***

```
std::cout << "==================================" << std::endl;
std::cout << "Sizes in bytes:" << std::endl;
std::cout << "float: " << sizeof(float) << std::endl;
std::cout << "double: " << sizeof(double) << std::endl;
std::cout << "long double: " << sizeof(long double) << std::endl;
std::cout << "==================================" << std::endl;
std::cout << std::endl;
```

```
==================================
Sizes in bytes:
float: 4
double: 8
long double: 16
==================================
```

Maszynowy epsilon

- Ustalamy początkową wartość *ε, a następnie dzielimy ją przez 2 do momentu, aż 1 + ε == 1*
- *Dzielenie przez 2 można traktować jako przesunięcie bitowe, co jest przydatne w przypadku obliczania ilości bitów dla mantysy*

```
float float_epsilon = 1.0f, float_tmp_epsilon = 1.0f;
while (1.0f + float_tmp_epsilon != 1.0f)
{
    float_epsilon = float_tmp_epsilon;
    float_tmp_epsilon /= 2.0f;
}
```

```
==================================
Machine epsilon:
float: 1.19209e-07
double: 2.22045e-16
long double: 1.0842e-19
==================================
```

Rozmiar mantysy

- Sposób analogiczny do wyznaczania maszynowego epsilonu, z różnicą sumowania bitów przy każdej operacji
- Wynik porównany ze stałą biblioteczną (podaną w nawiasie)

```
int float_mantissa_bits;
float f = 1.0f;
for (float_mantissa_bits = 0; 1.0f + f != 1.0f; ++float_mantissa_bits)
{
    f /= 2;
}
```

```
==================================
Mantissa size in bits:
float: 24(24)
double: 53(53)
long double: 64(64)
==================================
```

Rozmiar cechy

- Obliczony ze wzoru $\log_2(MAX_{EXP} - MIN_{EXP} + 1)$

```cpp
int float_exponent_bits = std::ceil(std::log2(FLT_MAX_EXP - FLT_MIN_EXP + 1));
int double_exponent_bits = std::ceil(std::log2(DBL_MAX_EXP - DBL_MIN_EXP + 1));
int long_double_exponent_bits = std::ceil(std::log2(LDBL_MAX_EXP - LDBL_MIN_EXP + 1));

std::cout << "==================================" << std::endl;
std::cout << "Exponent bits:" << std::endl;
std::cout << "float: " << float_exponent_bits << std::endl;
std::cout << "double: " << double_exponent_bits << std::endl;
std::cout << "long double: " << long_double_exponent_bits << std::endl;
std::cout << "==================================" << std::endl;
std::cout << std::endl;
```

```
==================================
Exponent bits:
float: 8
double: 11
long double: 15
==================================
```

Wartości 0

- Otrzymywane przez dzielenie 1 oraz -1 przez wielką liczbę

```cpp
float positive_zero = 1.0f / 100000000000000000000000000000000000000000000000000.0;
float negative_zero = -1.0f / 100000000000000000000000000000000000000000000000000.0;

std::cout << "====================================" << std::endl;
std::cout << "Zero check:" << std::endl;
std::cout << "Positive: " << positive_zero << "  | Sign bit: " << signbit(positive_zero) << std::endl;
std::cout << "Negative: " << negative_zero << " | Sign bit: " << signbit(negative_zero) << std::endl;
std::cout << "-0 == 0 -> " << (positive_zero == negative_zero) << std::endl;
std::cout << "====================================" << std::endl;
std::cout << std::endl;
```

```
====================================
Zero check:
Positive: 0  | Sign bit: 0
Negative: -0 | Sign bit: 1
-0 == 0 -> 1
====================================
```

Wartości inf

- Otrzymane przez podzielenie wartości 1 oraz -1 przez 0

```
std::cout << "==================================" << std::endl;
std::cout << "Infinity check:" << std::endl;
std::cout << "Positive infinity: " << (1 / 0.0) << std::endl;
std::cout << "Negative infinity: " << (-1 / 0.0) << std::endl;
std::cout << "-inf < FLT_MIN -> " << ((-1 / 0.0) < FLT_MIN) << std::endl;
std::cout << "==================================" << std::endl;
std::cout << std::endl;
```

```
==================================
Infinity check:
Positive infinity: inf
Negative infinity: -inf
-inf < FLT_MIN -> 1
==================================
```

Wartość NaN

- Otrzymana w wyniku operacji $\sqrt{(-2)}$
- NaN != NaN

```
float a = sqrt(-2);

std::cout << "==================================" << std::endl;
std::cout << "NaN check:" << std::endl;
std::cout << "Built-in NaN: " << NAN << std::endl;
std::cout << "a = sqrt(-2) = " << a << std::endl;
std::cout << "a == a -> " << (a == a) << std::endl;
std::cout << "isnan(a) -> " << isnan(a) << std::endl;
std::cout << "==================================" << std::endl;
std::cout << std::endl;
```

```
==================================
NaN check:
Built-in NaN: nan
a = sqrt(-2) = nan
a == a -> 0
isnan(a) -> 1
==================================
```

Porównania

Porównanie systemów

- Kompilator: g++
- Systemy: Windows 11 oraz Ubuntu 22.04 LTS

g++	Windows 11			Ubuntu 22.04 LTS		
Typ danych	float	double	long double	float	double	long double
Rozmiar (B)	4	8	16	4	8	16
Rozmiar mantysy (b)	24	53	64	24	53	64
Rozmiar cechy (b)	8	11	15	8	11	15
Maszynowy epsilon	1.19209e-07	2.22045e-16	1.0842e-19	1.19209e-07	2.22045e-16	1.0842e-19
Występowanie +0/-0	tak / tak			tak / tak		
Występowanie +inf/-inf	tak / tak			tak / tak		
Wynik sqrt(-2)	nan			-nan		

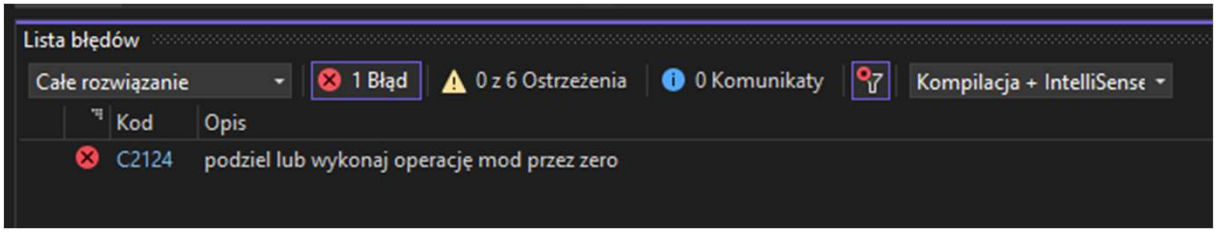
Porównanie kompilatorów

- Kompilatory: g++ oraz Microsoft Visual C++
- System: Windows 11

Windows 11	g++			MSVC		
Typ danych	float	double	long double	float	double	long double
Rozmiar (B)	4	8	16	4	8	8
Rozmiar mantysy (b)	24	53	64	24	53	53
Rozmiar cechy (b)	8	11	15	8	11	11
Maszynowy epsilon	1.19209e-07	2.22045e-16	1.0842e-19	1.19209e-07	2.22045e-16	2.22045e-16
Występowanie +0/-0	tak / tak			tak / tak		
Występowanie +inf/-inf	tak / tak			tak / tak		
Wynik sqrt(-2)	nan			-nan(ind)		

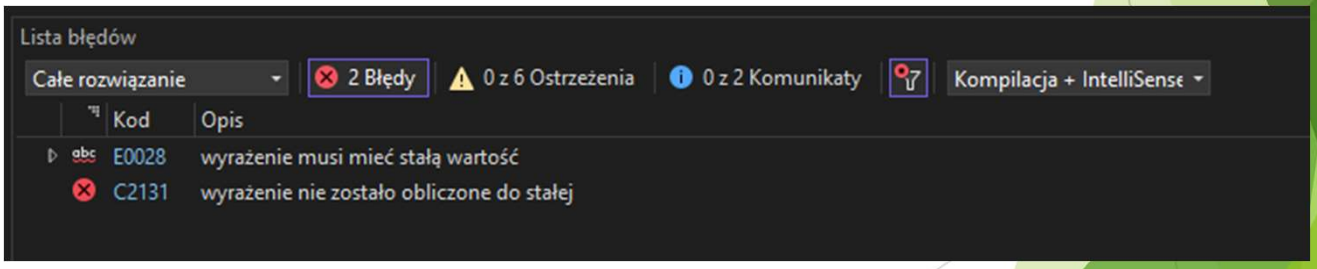
Restrykcyjność MSVC

- Dzielenie przez 0



- Deklarowanie rozmiaru tablicy jako zmiennej

```
int n = 10;
int arr[n];
```



Porównanie architektur

- Systemy:
  - Ubuntu 22.04 LTS 64-bit
  - Ubuntu 14.04 LTS 32-bit

Ubuntu	22.04 LTS 64-bit			14.04 LTS 32-bit		
Typ danych	float	double	long double	float	double	long double
Rozmiar (B)	4	8	16	4	8	12
Rozmiar mantysy (b)	24	53	64	24	53	64
Rozmiar cechy (b)	8	11	15	8	11	15
Maszynowy epsilon	1.19209e-07	2.22045e-16	1.0842e-19	1.0842e-19	1.0842e-19	1.0842e-19
Występowanie +0/-0	tak / tak			tak / tak		
Występowanie +inf/-inf	tak / tak			tak / tak		
Wynik sqrt(-2)	-nan			-nan		

Dziękuję za uwagę

Żródła:

- https://www.geeksforgeeks.org/ieee-full-form/
- https://en.wikipedia.org/wiki/IEEE_754
- https://stackoverflow.com/questions/502022/how-to-find-mantissa-length-on-a-particular-machine
- https://www.codeproject.com/Articles/824516/Concept-of-NaN-IND-INF-and-DEN