

# Informatyka 1

---

Politechnika Białostocka - Wydział Elektryczny  
Elektrotechnika, semestr II, studia stacjonarne I stopnia  
Rok akademicki 2018/2019

**Wykład nr 9 (10.05.2019)**

dr inż. Jarosław Forenc

## Plan wykładu nr 9

- Język C - tablice jednowymiarowe (wektory)
  - deklaracja, odwołania do elementów, inicjalizacja tablicy
  - generator liczb pseudolosowych
  - operacje na wektorze
  
- Standard IEEE 754
  - liczby 64-bitowe
  - zakres i precyzja liczb
  - wartości specjalne

## Język C - operacje na dużej ilości danych

```
#include <stdio.h>

int main(void)
{
    double U1, U2, U3, U4, U5;
    double I1, I2, I3, I4, I5;
    double R1, R2, R3, R4, R5;

    U1 = 5.0;
    U2 = 10.0;
    U3 = 15.0;
    U4 = 20.0;
    U5 = 25.0;

    I1 = 0.16;
    I2 = 0.21;
    I3 = 0.27;
    I4 = 0.33;
    I5 = 0.36;
```

## Język C - operacje na dużej ilości danych

```
R1 = U1/I1;  
R2 = U2/I2;  
R3 = U3/I3;  
R4 = U4/I4;  
R5 = U5/I5;  
  
printf("R1 = %f\n", R1);  
printf("R2 = %f\n", R2);  
printf("R3 = %f\n", R3);  
printf("R4 = %f\n", R4);  
printf("R5 = %f\n", R5);  
  
return 0;  
}
```

```
R1 = 31.250000  
R2 = 47.619048  
R3 = 55.555556  
R4 = 60.606061  
R5 = 69.444444
```

## Język C - operacje na dużej ilości danych (tablica)

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
    double U[5] = { 5.0, 10.0, 15.0, 20.0, 25.0 };
```

```
    double I[5] = { 0.16, 0.21, 0.27, 0.33, 0.36 };
```

```
    double R[5];
```

```
    int i;
```

```
    for (i=0; i<5; i++)  
        R[i] = U[i]/I[i];
```

```
    for (i=0; i<5; i++)  
        printf("R%d = %f\n", i+1, R[i]);
```

```
    return 0;
```

```
}
```

R1 = 31.250000

R2 = 47.619048

R3 = 55.555556

R4 = 60.606061

R5 = 69.444444

	0	1	2	3	4
U	5.0	10.0	15.0	20.0	25.0
I	0.16	0.21	0.27	0.33	0.36
R	31.25	47.62	55.56	60.61	69.44

## Język C - tablica elementów

- **Tablica** - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu

wektor

5	3	-2	1	-4
---	---	----	---	----

macierz

a	c	d	m
p	d	q	l
a	t	x	v

1.2	2.5	2.0	10.0
-0.1	4.3	6.2	-5.1
0.0	12.2	4.1	-2.2

## Język C - tablica jednowymiarowa

- **Tablica** - ciągły obszar pamięci, w którym umieszczone są elementy tego samego typu
- **Wektor** - tablica jednowymiarowa

5	3	-2	0	-4
---	---	----	---	----

- liczby całkowite

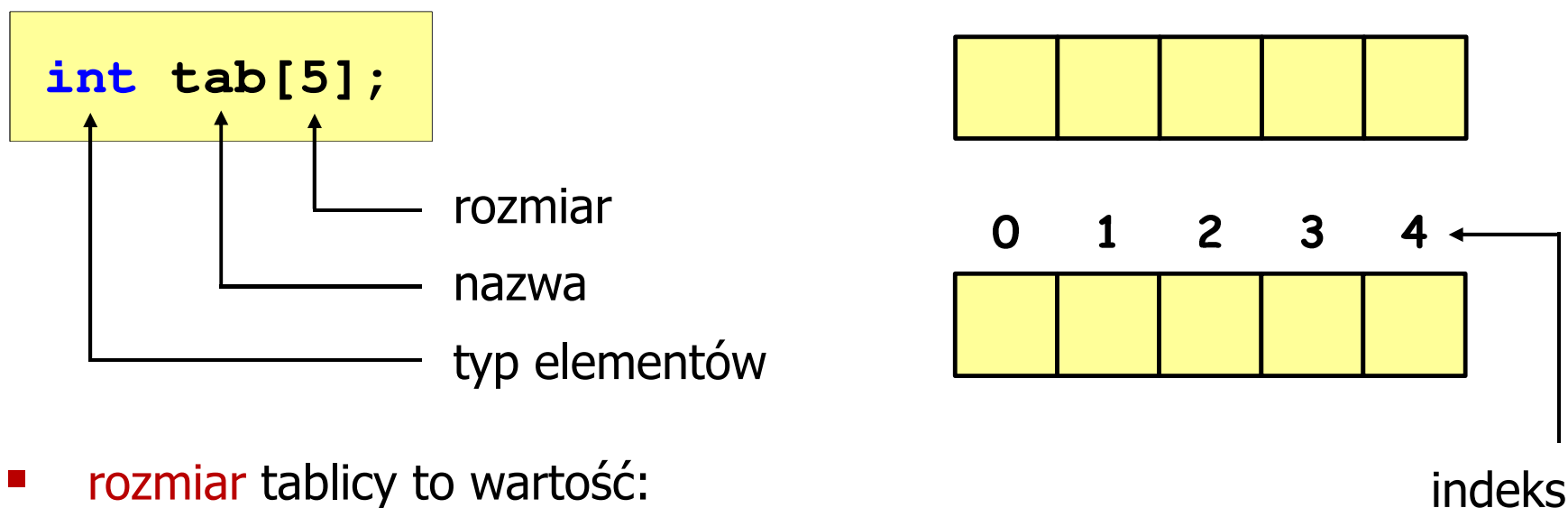
3.1	0.2	2.3	-1.3	1.5	1.1	-4.0
-----	-----	-----	------	-----	-----	------

- liczby rzeczywiste

a	Z	x	&	M	+
---	---	---	---	---	---

- znaki

## Język C - deklaracja tablicy jednowymiarowej



- **rozmiar** tablicy to wartość:
  - całkowita, dodatnia
  - znana na etapie kompilacji programu  
(stała liczbowa: `5`, `#define N 5`, `const int n = 5;`)

```
int tab[5];
```

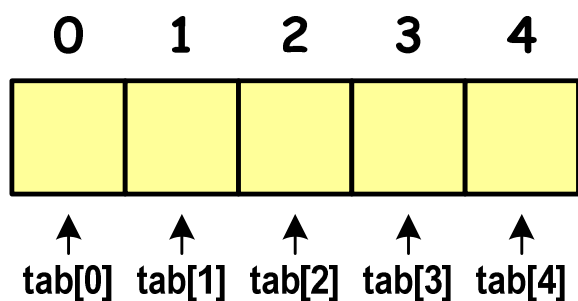
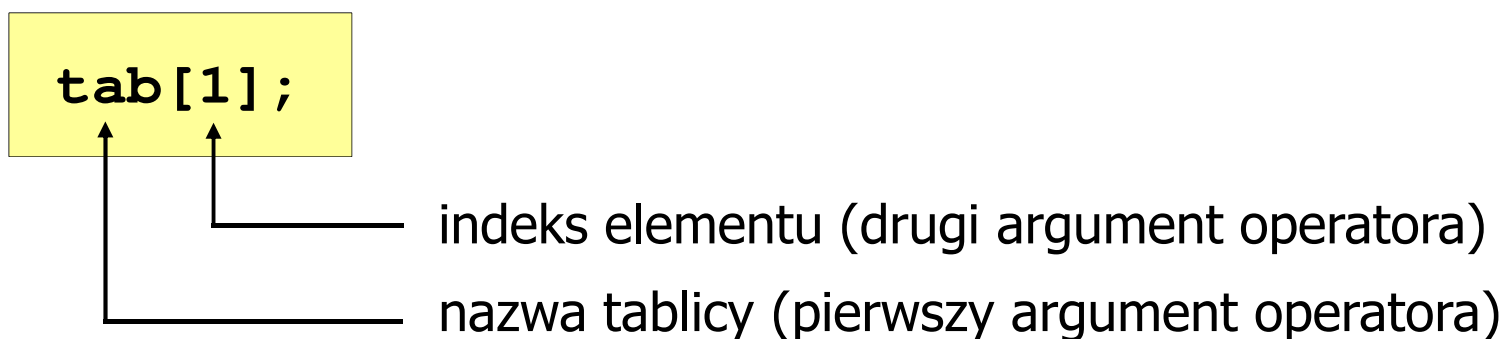
```
int tab[N];
```

```
int tab[n];
```



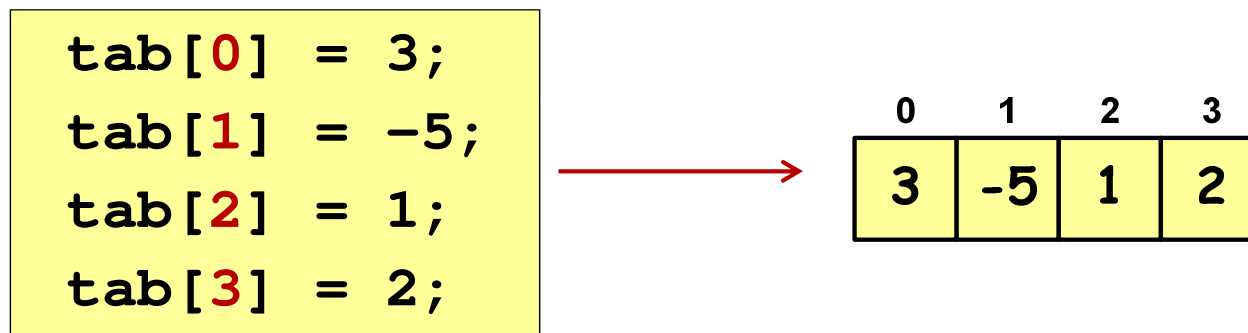
## Język C - odwołania do elementów tablicy

[ ] - dwuargumentowy operator indeksowania



- indeks:
  - stała liczbowa, np. 0, 1, 10
  - nazwa zmiennej, np. i, idx
  - wyrażenie, np.  $i*j+5$

## Język C - odwołania do elementów tablicy



- Każdy element tablicy traktowany jest tak samo jak zmienna typu `int`

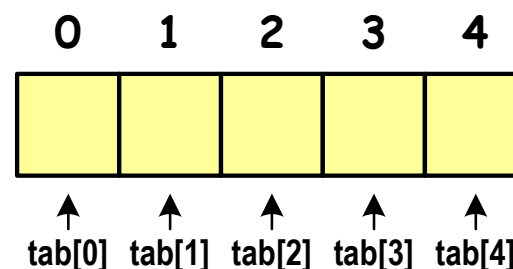
```
printf("%d", tab[0]);
```

```
scanf("%d", &tab[1]);
```

## Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

```
int tab[5];  
tab[5] = 10;
```



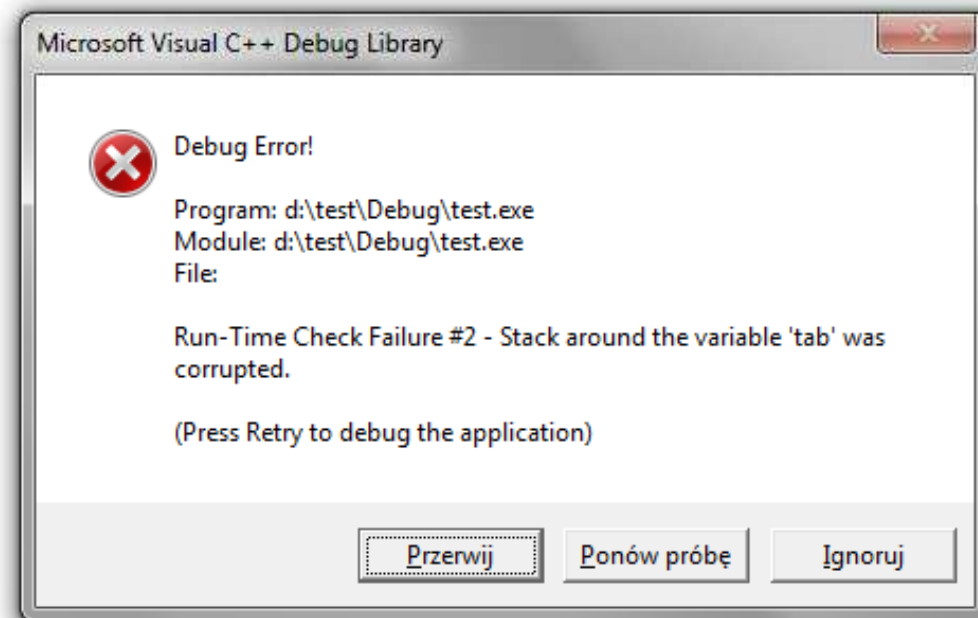
- **błąd!!!** - nie istnieje element **tab[5]**

- Kompilator nie zasygnalizuje błędu
- Program wykona operację
- Środowisko programistyczne może zasygnalizować problem

## Język C - odwołania do elementów tablicy

- Przy odwołaniach do elementów tablicy kompilator nie sprawdza poprawności indeksów

```
int tab[5];  
tab[5] = 10;
```



## Język C - inicjalizacja tablicy jednowymiarowej

```
int tab[5] = {1,2,3,4,5};
```

0	1	2	3	4
1	2	3	4	5

```
int tab[5] = {1,2,3};
```

0	1	2	3	4
1	2	3	0	0

```
int tab[5] = {1,2,3,4,5,6};
```

- błąd kompilacji

```
int tab[] = {1,2,3,4,5};
```

0	1	2	3	4
1	2	3	4	5

## Język C - odwołania do elementów tablicy

- Zapisanie wartości **1** do wszystkich elementów tablicy

```
int tab[5];
```

```
tab[0] = 1;
```

```
tab[1] = 1;
```

```
tab[2] = 1;
```

```
tab[3] = 1;
```

```
tab[4] = 1;
```

0	1	2	3	4
1	1	1	1	1

```
int tab[5], i;
```

```
for (i=0; i<5; i++)
```

```
    tab[i] = 1;
```

## Język C - generator liczb pseudolosowych

- `rand()` - zwraca liczbę pseudolosową - zakres: `0 ... 32767`
- `srand()` - inicjalizuje generator liczb pseudolosowych
- Plik nagłówkowy: `stdlib.h` (`time.h`)

```
int x, y;
srand( (unsigned int) time(NULL) );
x = rand();           // zakres <0, 32767>
y = rand() % 100;    // zakres <0, 99>
```

## Język C - operacje na wektorze

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define N 10

int main(void)
{
    int tab[N], i;

    /* generowanie elementów tablicy */

    srand((unsigned int) time(NULL));

    for (i=0; i<N; i++)
        tab[i] = rand() % 20;
```

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8



## Język C - operacje na wektorze

```
/* wyświetlenie elementów tablicy */  
  
printf("Elementy tablicy:\n");  
for (i=0; i<N; i++)  
    printf("%d  ", tab[i]);  
printf("\n");
```

Elementy tablicy:

7 12 1 16 1 11 14 5 19 8

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

**N = 10**

## Język C - operacje na wektorze

```
/* wyświetlenie elementów w odwrotnej kolejności */  
  
printf("Elementy w odwrotnej kolejności:\n");  
for (i=N-1; i>=0; i--)  
    printf("%d  ", tab[i]);  
printf("\n");
```

```
Elementy w odwrotnej kolejności:  
8  19  5  14  11  1  16  1  12  7
```

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

**N = 10**

## Język C - operacje na wektorze

```
/* wyszukanie elementu o najmniejszej wartości */  
  
int min;  
  
min = tab[0];  
for (i=1; i<N; i++)  
    if (tab[i]<min)  
        min = tab[i];  
printf("Wartosc elementu najmniejszego: %d\n",min);
```

Wartosc elementu najmniejszego: 1

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

N = 10

## Język C - operacje na wektorze

```
/* indeksy elementów o najmniejszej wartości */  
  
printf("Indeksy elementu najmniejszego: ");  
for (i=0; i<N; i++)  
    if (tab[i]==min)  
        printf("%d ", i);  
printf("\n");
```

Indeksy elementu najmniejszego: 2 4

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

N = 10

## Język C - operacje na wektorze

```
/* suma i średnia arytmetyczna elementów tablicy */  
  
int suma = 0;  
float srednia;  
  
for (i=0; i<N; i++)  
    suma = suma + tab[i];  
srednia = (float) suma/N;  
printf("Suma: %d, srednia: %g\n", suma, srednia);
```

Suma: 94, srednia: 9.4

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

**N = 10**

## Język C - operacje na wektorze

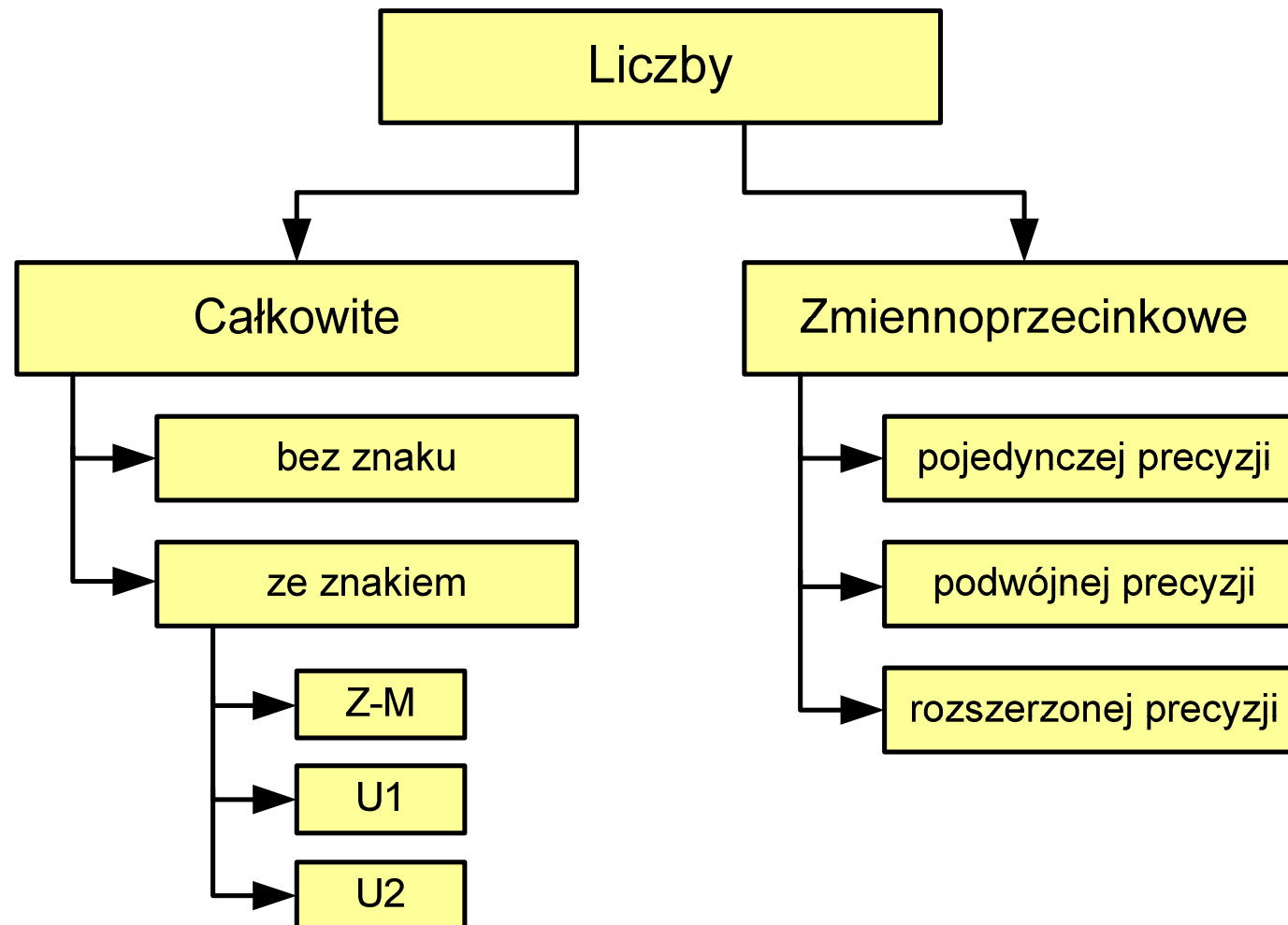
```
/* liczba parzystych elementów tablicy */  
  
int ile = 0;  
  
for (i=0; i<N; i++)  
    if (tab[i]%2==0)  
        ile++;  
printf("Liczba parzystych elementow: %d\n",ile);
```

Liczba parzystych elementow: 4

0	1	2	3	4	5	6	7	8	9
7	12	1	16	1	11	14	5	19	8

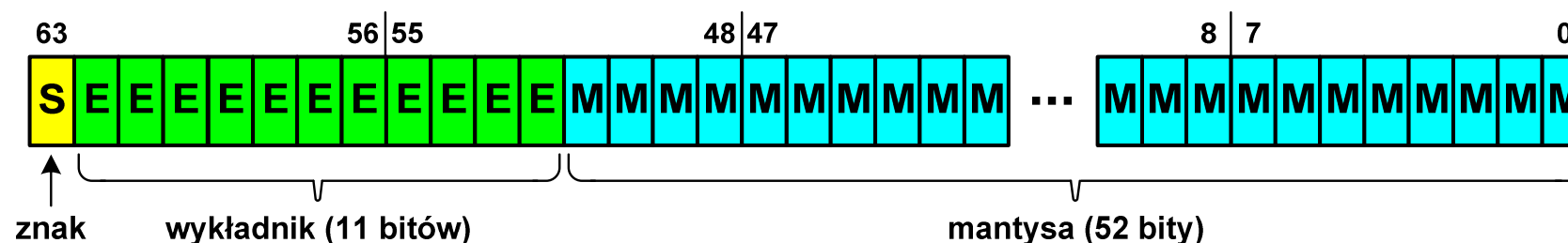
**N = 10**

# Reprezentacja liczb w systemach komputerowych



## Standard IEEE 754 - liczby 64-bitowe

- Liczba podwójnej precyzji przechowywana jest na 64 bitach:



- Pierwszy bit w zapisie (bit nr 63) jest **bitem znaku** (0 - liczba dodatnia, 1 - liczba ujemna)
- **Wykładnik** zapisywany jest na **11 bitach** (bity nr 62-52) z nadmiarem o wartości 1023
- **Wykładnik** może przyjmować wartości od -1023 (wszystkie bity wyzerowane) do 1024 (wszystkie bity ustawione na 1)
- **Mantysa** zapisywana jest na 52 bitach (pierwszy bit mantysy, zawsze równy 1, nie jest zapamiętywany)



## Standard IEEE 754 - zakres liczb

### ■ Pojedyncza precyzja:

- największa wartość:  $\approx 3,4 \cdot 10^{38}$
- najmniejsza wartość:  $\approx 1,4 \cdot 10^{-45}$
- zakres liczb:  $\langle -3,4 \cdot 10^{38} \dots -1,4 \cdot 10^{-45} \rangle \cup \{0\} \cup \langle 1,4 \cdot 10^{-45} \dots 3,4 \cdot 10^{38} \rangle$

### ■ Podwójna precyzja:

- największa wartość:  $\approx 1,8 \cdot 10^{308}$
- najmniejsza wartość:  $\approx 4,9 \cdot 10^{-324}$
- zakres liczb:  $\langle -1,8 \cdot 10^{308} \dots -4,9 \cdot 10^{-324} \rangle \cup \{0\} \cup \langle 4,9 \cdot 10^{-324} \dots 1,8 \cdot 10^{308} \rangle$

### ■ Podwójna rozszerzona precyzja:

- największa wartość:  $\approx 1,2 \cdot 10^{4932}$
- najmniejsza wartość:  $\approx 3,6 \cdot 10^{-4951}$
- zakres liczb:  $\langle -1,2 \cdot 10^{4932} \dots -3,6 \cdot 10^{-4951} \rangle \cup \{0\} \cup \langle 3,6 \cdot 10^{-4951} \dots 1,2 \cdot 10^{4932} \rangle$

## Standard IEEE 754 - precyzja liczb

- **Precyzja** - liczba zapamiętywanych cyfr znaczących w systemie (10)

4,86452137846 → 4,864521 - 7 cyfr znaczących

- Precyzja liczby zależy od **liczby bitów mantysy**
- Liczba bitów potrzebnych do zakodowania **1** cyfry dziesiętnej:

$$10^1 = 2^n \rightarrow n = \log_2(10) \approx 3,321928$$

- Liczba cyfr dziesiętnych (**d**) możliwa do zakodowania na **m** bitach:

$\log_2(10)$  bitów - **1** cyfra dziesiętna  
**m** bitów - **d** cyfr dziesiętnych

$$d = \frac{m}{\log_2(10)}$$

## Standard IEEE 754 - precyzja liczb

- Dla formatu pojedynczej precyzji:

- mantysa:  $23 + 1 = 24$  bity

- cyfry znaczące: 7

$$d = \frac{24}{\log_2(10)} = \frac{24}{3,321928} = 7,2247 \approx 7$$

- Dla formatu podwójnej precyzji:

- mantysa:  $52 + 1 = 53$  bity

- cyfry znaczące: 16

$$d = \frac{53}{\log_2(10)} = \frac{53}{3,321928} = 15,9546 \approx 16$$

- Dla formatu podwójnej rozszerzonej precyzji:

- mantysa:  $63 + 1 = 64$  bity

- cyfry znaczące: 19

$$d = \frac{64}{\log_2(10)} = \frac{64}{3,321928} = 19,2659 \approx 19$$

## Standard IEEE 754 - precyzja liczb

```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    float x;
```

```
    double y;
```

```
    x = 1234567890.0;    /* 1.234.567.890 */
```

```
    y = 1234567890.0;    /* 1.234.567.890 */
```

```
    printf("float -> %f\n", x);
```

```
    printf("double -> %f\n\n", y);
```

```
    y = 12345678901234567890.0;
```

```
    printf("double -> %f\n", y);
```

```
    return 0;
```

```
}
```

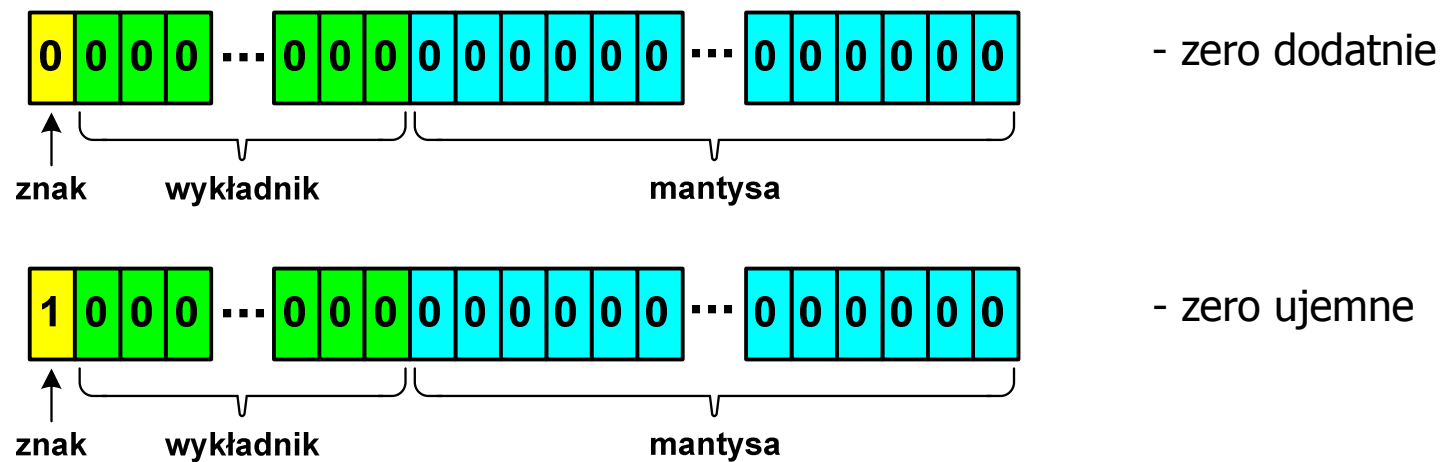
```
float -> 1234567936.000000
```

```
double -> 1234567890.000000
```

```
double -> 12345678901234567000.000000
```

## Standard IEEE 754 - wartości specjalne

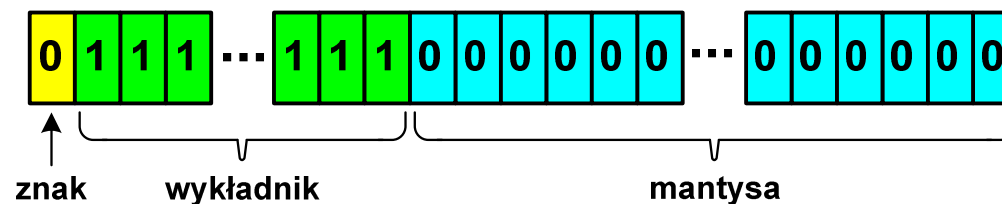
### ■ Zero:



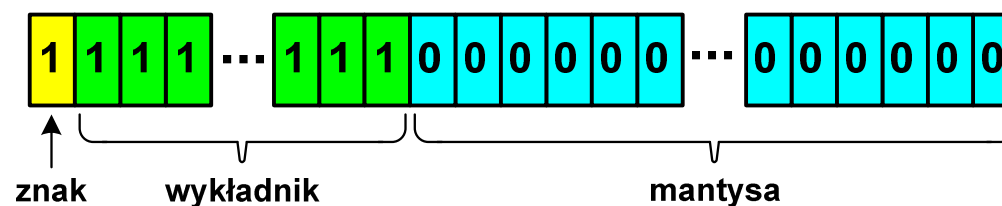
- Podczas porównań zero dodatnie i ujemne są traktowane jako równe sobie

## Standard IEEE 754 - wartości specjalne

### ■ Nieskończoność:



- nieskończoność dodatnia

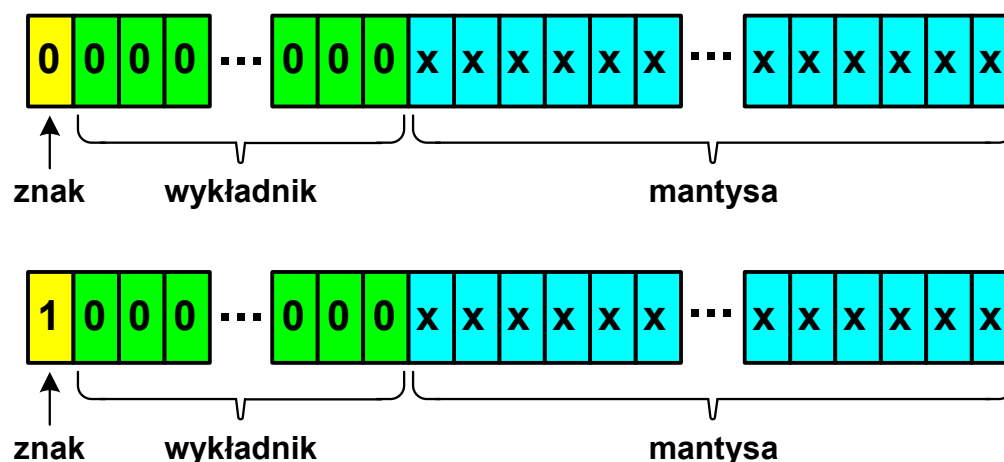


- nieskończoność ujemna

- Nieskończoność występuje w przypadku wystąpienia **nadmiaru** (przepełnienia) oraz przy dzieleniu przez zero

## Standard IEEE 754 - wartości specjalne

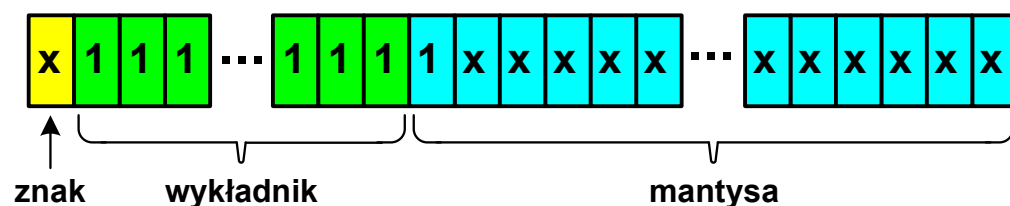
### ■ Liczba zdenormalizowana:



- Pojawia się, gdy występuje **niedomiar** (ang. **underflow**), ale wynik operacji można jeszcze zapisać denormalizując mantysę
- Mantysa nie posiada domyślnej części całkowitej równej **1**, tzn. reprezentuje liczbę o postaci **0,xxx...xxx**, a nie **1,xxx...xxx**

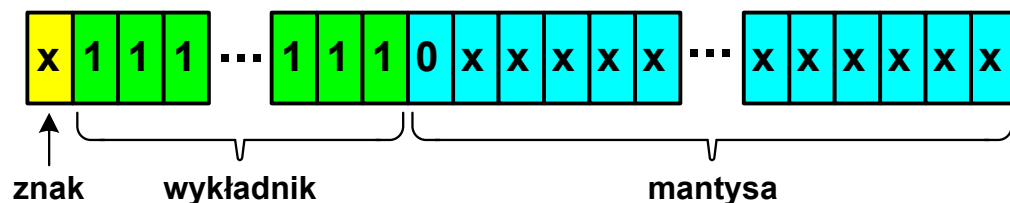
## Standard IEEE 754 - wartości specjalne

- **Nieliczby - NaN (Not A Number)** - nie reprezentują wartości liczbowej
- Powstają w wyniku wykonania niedozwolonej operacji
- **QNaN (ang. Quiet NaN)** - ciche nieliczby



- „przechodzą” przez działania arytmetyczne (brak przerwania wykonywania programu)

- **SNaN (ang. Signaling NaN)** - sygnalizujące, istotne, głośne nieliczby



- zgłoszenie wyjątku (przerwanie wykonywania programu)



Koniec wykładu nr 9

Dziękuję za uwagę!