

**Stefan Sokołowski**

# **Kiedy i dlaczego komputery kłamią**

**— paradoksy arytmetyki komputerowej**

**Inst. Informatyki Stosowanej, ANS**

**Elbląg, 19 maja 2022**

# Buty siedmiomilowe

**MARZENIE:**



# Buty siedmiomilowe

**MARZENIE:**



**SPEŁNIENIE:**



# Buty siedmiomilowe

**MARZENIE:**



**SPEŁNIENIE:**



... ale:

- samochody potrzebują budowy dróg, parkingów, stacji obsługi

# Buty siedmiomilowe

## MARZENIE:



## SPEŁNIENIE:



... ale:

- samochody potrzebują budowy dróg, parkingów, stacji obsługi, ...
- samochody powodują wypadki

# Buty siedmiomilowe

## MARZENIE:



## SPEŁNIENIE:



... ale:

- samochody potrzebują budowy dróg, parkingów, stacji obsługi, ...
- samochody powodują wypadki,
- samochody rozleniwiają ludzi,
- itd.

# Latać jak ptak

## MARZENIE:

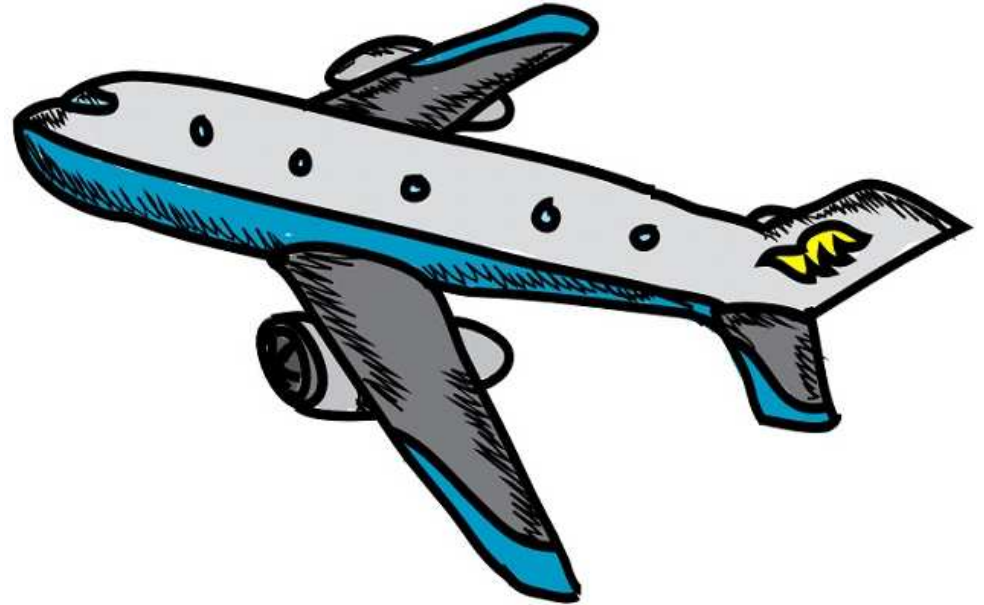


# Latać jak ptak

**MARZENIE:**



**SPEŁNIENIE:**

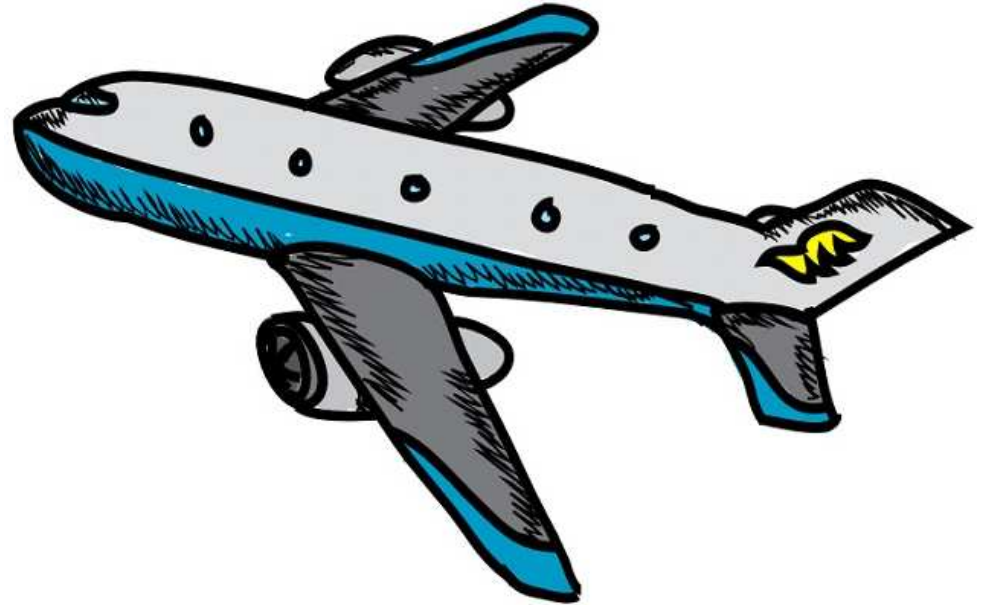


# Latać jak ptak

**MARZENIE:**



**SPEŁNIENIE:**



... ale:

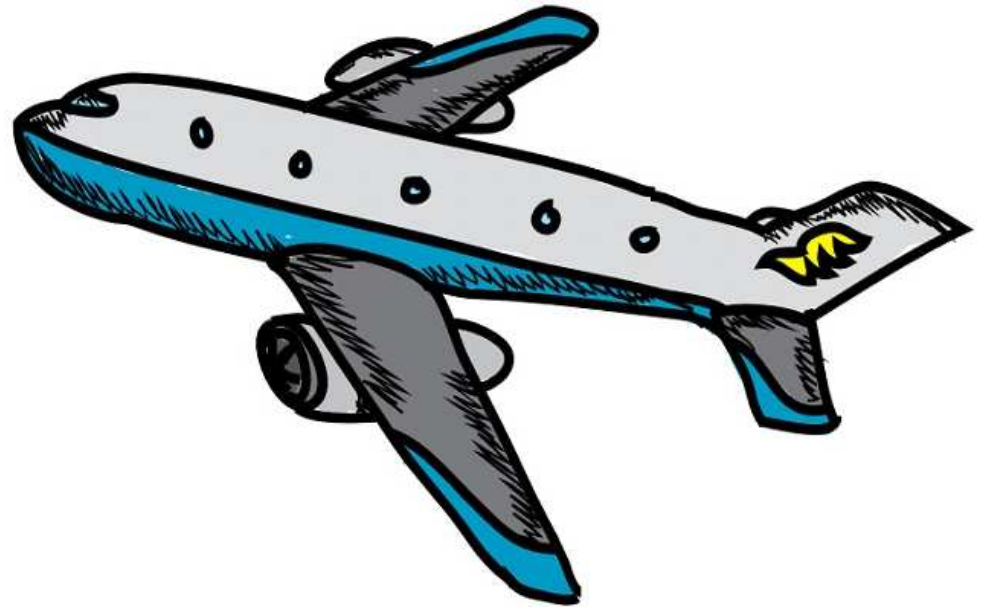
- samoloty potrzebują wielkich lotnisk

# Latać jak ptak

## MARZENIE:



## SPEŁNIENIE:



... ale:

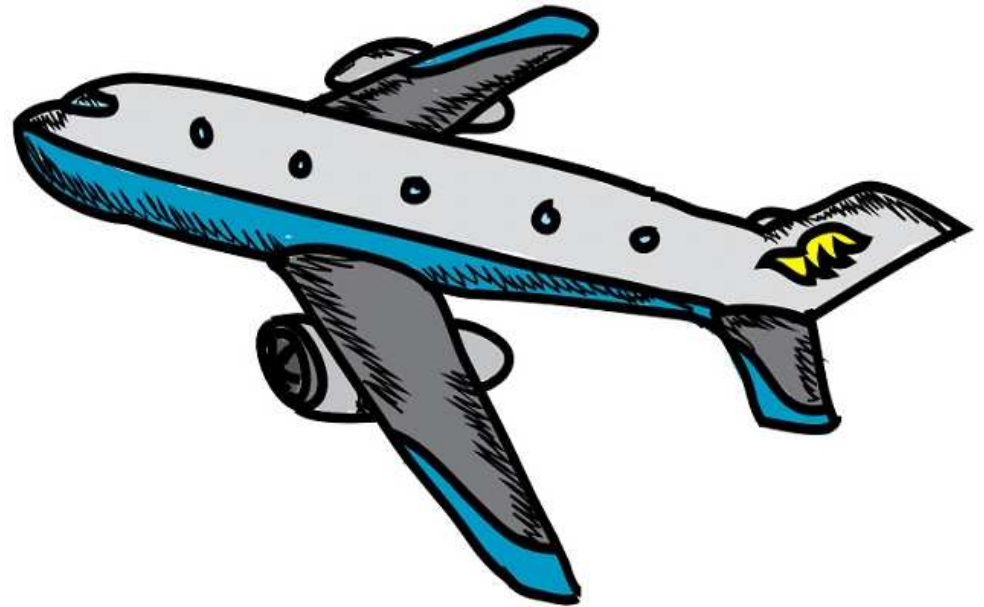
- samoloty potrzebują wielkich lotnisk,
- samoloty potrzebują kontroli ruchu lotniczego

# Latać jak ptak

## MARZENIE:



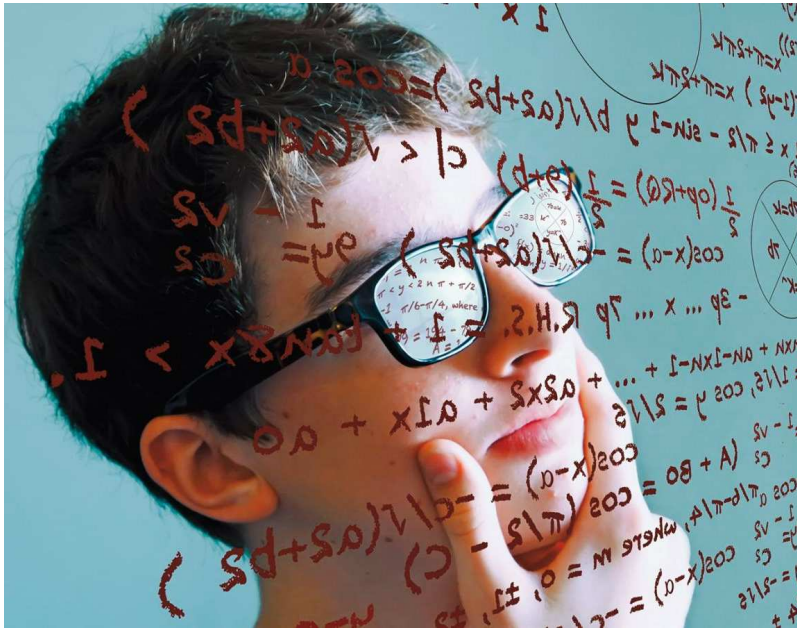
## SPEŁNIENIE:



... ale:

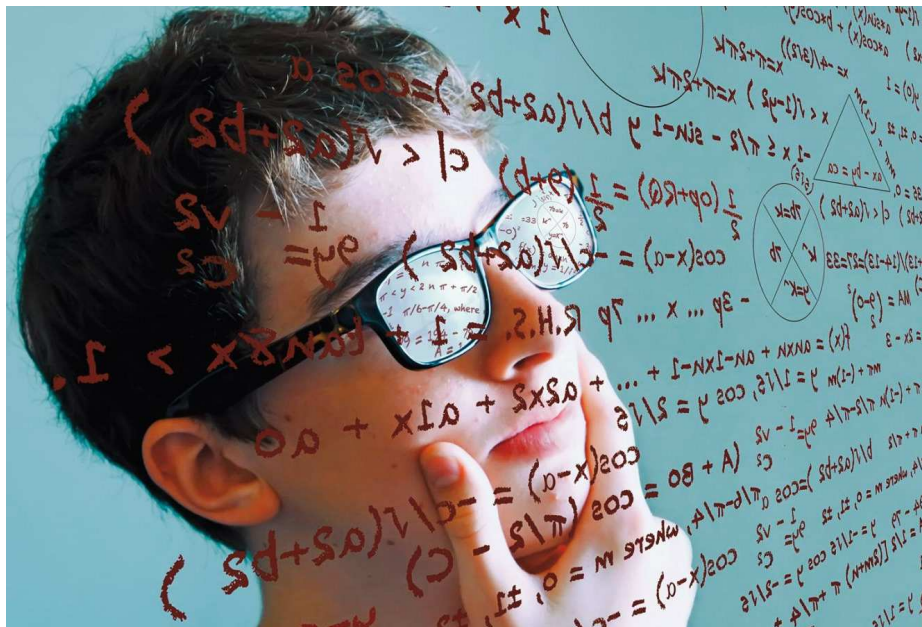
- samoloty potrzebują wielkich lotnisk,
- samoloty potrzebują kontroli ruchu lotniczego,
- samoloty zużywają nieodnawialne zasoby Ziemi,
- itd.

## MARZENIE:



# Mądrość

## MARZENIE:

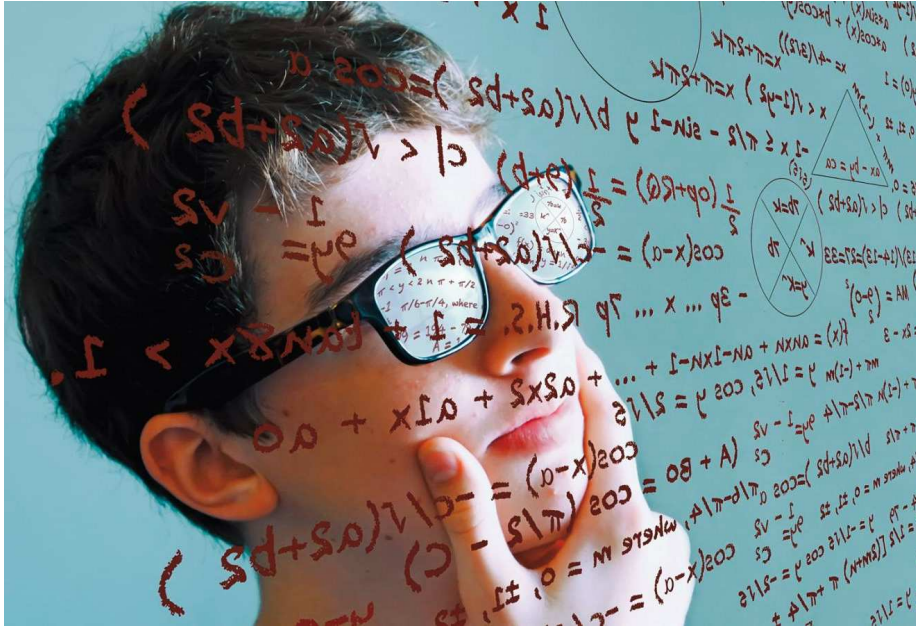


## SPEŁNIENIE:



# Mądrość

## MARZENIE:



## SPEŁNIENIE:



... ale:

**KOMPUTERY KŁAMIA!**

# Komputer kłamie (liczby całkowite)

# Komputer kłamie (liczby całkowite)

Program:

```
#include<stdio.h>
int main() {
    int n=50000;
    printf("\n  %i * %i == %i\n\n", n, n, n*n);
    return 0;
}
```

# Komputer kłamie (liczby całkowite)

Program:

```
#include<stdio.h>
int main() {
    int n=50000;
    printf("\n  %i * %i == %i\n\n", n, n, n*n);
    return 0;
}
```

wyświetla:

```
50000 * 50000 == -1794967296
```

# Komputer kłamie (liczby całkowite)

Program:

```
#include<stdio.h>
int main() {
    int n=50000;
    printf("\n  %i * %i == %i\n\n", n, n, n*n);
    return 0;
}
```

wyświetla:

```
50000 * 50000 == -1794967296
```

**Kwadrat dodatniej jest **ujemny**?!**

# Komputer kłamie (liczby rzeczywiste)

# Komputer kłamie (liczby rzeczywiste)

Program:

```
#include<stdio.h>
int main() {
    if ( 5 / 3.0 == 5 * (1 / 3.0) ) printf("\n  rowne \n\n");
    if ( 5 / 3.0 < 5 * (1 / 3.0) ) printf("\n  mniejsze \n\n");
    if ( 5 / 3.0 > 5 * (1 / 3.0) ) printf("\n  wieksze \n\n");
    return 0;
}
```

# Komputer kłamie (liczby rzeczywiste)

Program:

```
#include<stdio.h>
int main() {
    if ( 5 / 3.0 == 5 * (1 / 3.0) ) printf("\n  rowne \n\n");
    if ( 5 / 3.0 < 5 * (1 / 3.0) ) printf("\n  mniejsze \n\n");
    if ( 5 / 3.0 > 5 * (1 / 3.0) ) printf("\n  wieksze \n\n");
    return 0;
}
```

wyświetla:

wieksze

# Komputer kłamie (liczby rzeczywiste)

Program:

```
#include<stdio.h>
int main() {
    if ( 5 / 3.0 == 5 * (1 / 3.0) ) printf("\n  rowne \n\n");
    if ( 5 / 3.0 < 5 * (1 / 3.0) ) printf("\n  mniejsze \n\n");
    if ( 5 / 3.0 > 5 * (1 / 3.0) ) printf("\n  wieksze \n\n");
    return 0;
}
```

wyświetla:

wieksze

Dzielenie przez **3** daje **wiekszy** wynik  
niż mnożenie przez  $\frac{1}{3}$ ?!

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej  
[−2 mld . . 2 mld]

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej  
[−2 mld . . 2 mld]

rok świetlny

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej  
[−2 mld . . 2 mld]

$$\text{rok świetlny} = \text{rok} \cdot \text{prędkość światła}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej  
[−2 mld . . 2 mld]

$$\text{rok świetlny} = \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$[-2 \text{ mld} \dots 2 \text{ mld}]$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \end{aligned}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$[-2 \text{ mld} \dots 2 \text{ mld}]$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} \end{aligned}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$[-2 \text{ mld} \dots 2 \text{ mld}]$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$[-2 \text{ mld} \dots 2 \text{ mld}]$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

ZA DUŻO

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$[-2 \text{ mld} \dots 2 \text{ mld}]$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

min. roczny przychód 100 tys. ludzi

ZA DUŻO

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$$[-2 \text{ mld} \dots 2 \text{ mld}]$$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

ZA DUŻO

$$\text{min. roczny przychód 100 tys. ludzi} = 100\,000 \cdot 3010 \frac{\text{zł}}{\text{miesiąc}} \cdot \text{rok}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$$[-2 \text{ mld} \dots 2 \text{ mld}]$$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

ZA DUŻO

$$\begin{aligned} \text{min. roczny przychód 100 tys. ludzi} &= 100\,000 \cdot 3010 \frac{\text{zł}}{\text{miesiąc}} \cdot \text{rok} \\ &= 301\,000\,000 \frac{\text{zł}}{\text{miesiąc}} \cdot 12 \text{ miesięcy} \end{aligned}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$$[-2 \text{ mld} \dots 2 \text{ mld}]$$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

ZA DUŻO

$$\begin{aligned} \text{min. roczny przychód 100 tys. ludzi} &= 100\,000 \cdot 3010 \frac{\text{zł}}{\text{miesiąc}} \cdot \text{rok} \\ &= 301\,000\,000 \frac{\text{zł}}{\text{miesiąc}} \cdot 12 \text{ miesięcy} = 3\,612\,000\,000 \text{ zł} \end{aligned}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$$[-2 \text{ mld} \dots 2 \text{ mld}]$$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

ZA DUŻO

$$\begin{aligned} \text{min. roczny przychód 100 tys. ludzi} &= 100\,000 \cdot 3010 \frac{\text{zł}}{\text{miesiąc}} \cdot \text{rok} \\ &= 301\,000\,000 \frac{\text{zł}}{\text{miesiąc}} \cdot 12 \text{ miesięcy} = 3\,612\,000\,000 \text{ zł} \\ &\approx 3.6 \text{ miliarda zł} \end{aligned}$$

# Komputerowa arytmetyka liczb całkowitych

Działaniom w typie `int` można ufać w zakresie mniej więcej

$$[-2 \text{ mld} \dots 2 \text{ mld}]$$

$$\begin{aligned} \text{rok świetlny} &= \text{rok} \cdot \text{prędkość światła} \approx \text{rok} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx (365.25 \cdot 24 \cdot 60 \cdot 60) \text{ s} \cdot 300\,000 \frac{\text{km}}{\text{s}} \\ &\approx 9.5 \cdot 10^{12} \text{ km} = 9.5 \text{ biliona km} \end{aligned}$$

$$\begin{aligned} \text{min. roczny przychód 100 tys. ludzi} &= 100\,000 \cdot 3010 \frac{\text{zł}}{\text{miesiąc}} \cdot \text{rok} \\ &= 301\,000\,000 \frac{\text{zł}}{\text{miesiąc}} \cdot 12 \text{ miesięcy} = 3\,612\,000\,000 \text{ zł} \\ &\approx 3.6 \text{ miliarda zł} \end{aligned}$$

# Komputerowa arytmetyka liczb całkowitych

Program (silnia):

```
#include<stdio.h>
int main() {
    int  n=1, s=1;
    printf("\n");
    while (1) {
        printf("  %2i! == %10i", n, s);
        n++; s*=n; getchar();
    }
    return 0;
}
```

# Komputerowa arytmetyka liczb całkowitych

Program (silnia):

```
#include<stdio.h>
int main() {
    int n=1, s=1;
    printf("\n");
    while (1) {
        printf("  %2i! == %10i", n, s);
        n++; s*=n; getchar();
    }
    return 0;
}
```

wyświetla:

|        |            |
|--------|------------|
| 1! ==  | 1          |
| 2! ==  | 2          |
| 3! ==  | 6          |
| 4! ==  | 24         |
| 5! ==  | 120        |
| 6! ==  | 720        |
| 7! ==  | 5040       |
| 8! ==  | 40320      |
| 9! ==  | 362880     |
| 10! == | 3628800    |
| 11! == | 39916800   |
| 12! == | 479001600  |
| 13! == | 1932053504 |
| 14! == | 1278945280 |
| 15! == | 2004310016 |
| 16! == | 2004189184 |
| 17! == | -288522240 |

# Komputerowa arytmetyka liczb całkowitych

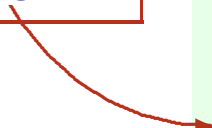
Program (silnia):

```
#include<stdio.h>
int main() {
    int n=1, s=1;
    printf("\n");
    while (1) {
        printf("  %2i! == %10i", n, s);
        n++; s*=n; getchar();
    }
    return 0;
}
```

wyświetla:

|        |            |
|--------|------------|
| 1! ==  | 1          |
| 2! ==  | 2          |
| 3! ==  | 6          |
| 4! ==  | 24         |
| 5! ==  | 120        |
| 6! ==  | 720        |
| 7! ==  | 5040       |
| 8! ==  | 40320      |
| 9! ==  | 362880     |
| 10! == | 3628800    |
| 11! == | 39916800   |
| 12! == | 479001600  |
| 13! == | 1932053504 |
| 14! == | 1278945280 |
| 15! == | 2004310016 |
| 16! == | 2004189184 |
| 17! == | -288522240 |

> 400 mln



# Komputerowa arytmetyka liczb całkowitych

Program (silnia):

```
#include<stdio.h>
int main() {
    int n=1, s=1;
    printf("\n");
    while (1) {
        printf("  %2i! == %10i", n, s);
        n++; s*=n; getchar();
    }
    return 0;
}
```

wyświetla:

|        |            |
|--------|------------|
| 1! ==  | 1          |
| 2! ==  | 2          |
| 3! ==  | 6          |
| 4! ==  | 24         |
| 5! ==  | 120        |
| 6! ==  | 720        |
| 7! ==  | 5040       |
| 8! ==  | 40320      |
| 9! ==  | 362880     |
| 10! == | 3628800    |
| 11! == | 39916800   |
| 12! == | 479001600  |
| 13! == | 1932053504 |
| 14! == | 1278945280 |
| 15! == | 2004310016 |
| 16! == | 2004189184 |
| 17! == | -288522240 |

> 400 mln

# Komputerowa arytmetyka liczb całkowitych

Program (silnia):

```
#include<stdio.h>
int main() {
    int n=1, s=1;
    printf("\n");
    while (1) {
        printf("  %2i! == %10i", n, s);
        n++; s*=n; getchar();
    }
    return 0;
}
```

wyświetla:

|        |            |
|--------|------------|
| 1! ==  | 1          |
| 2! ==  | 2          |
| 3! ==  | 6          |
| 4! ==  | 24         |
| 5! ==  | 120        |
| 6! ==  | 720        |
| 7! ==  | 5040       |
| 8! ==  | 40320      |
| 9! ==  | 362880     |
| 10! == | 3628800    |
| 11! == | 39916800   |
| 12! == | 479001600  |
| 13! == | 1932053504 |
| 14! == | 1278945280 |
| 15! == | 2004310016 |
| 16! == | 2004189184 |
| 17! == | -288522240 |

> 400 mln

silnia dodatniej **ujemna**?!

silnia większej niż 5 bez **zera** na końcu?!

# Cudowne uleczenie wyniku

2 mld



−2 mld

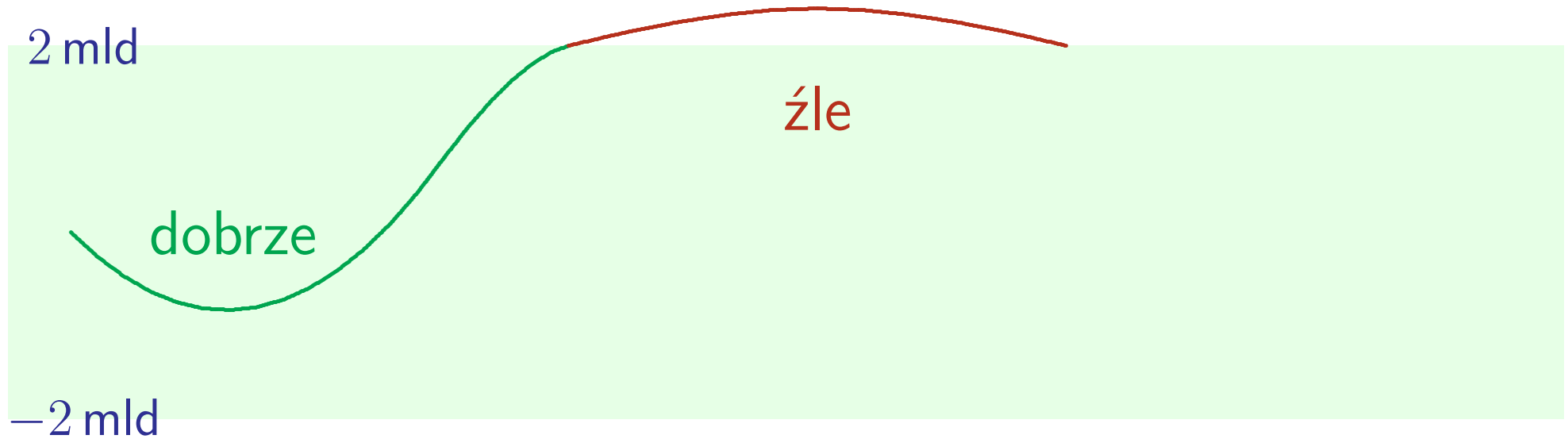
# Cudowne uleczenie wyniku



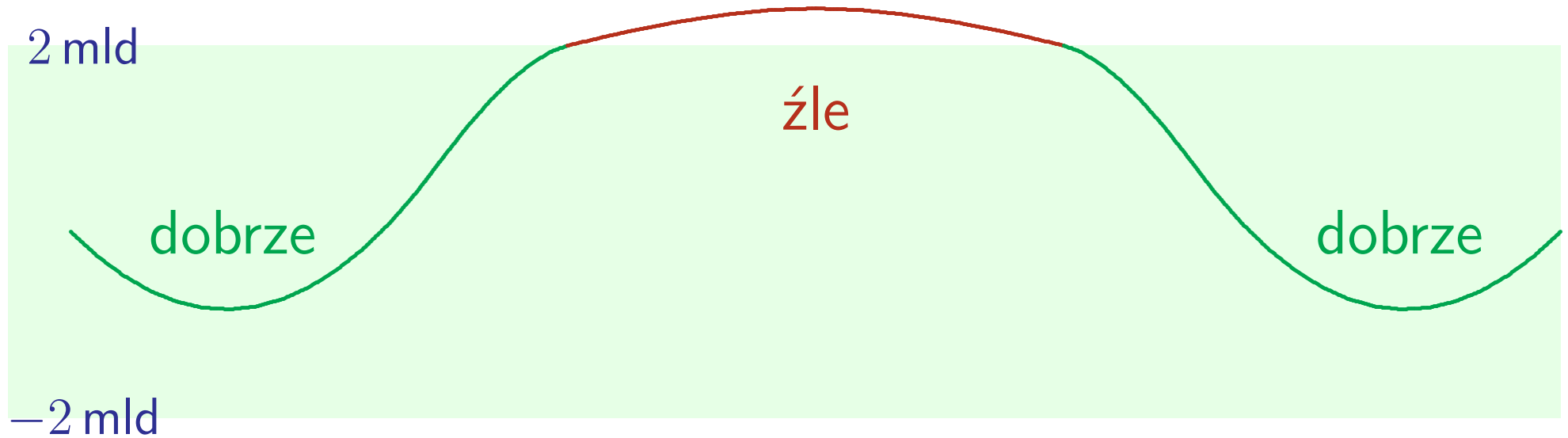
# Cudowne uleczenie wyniku



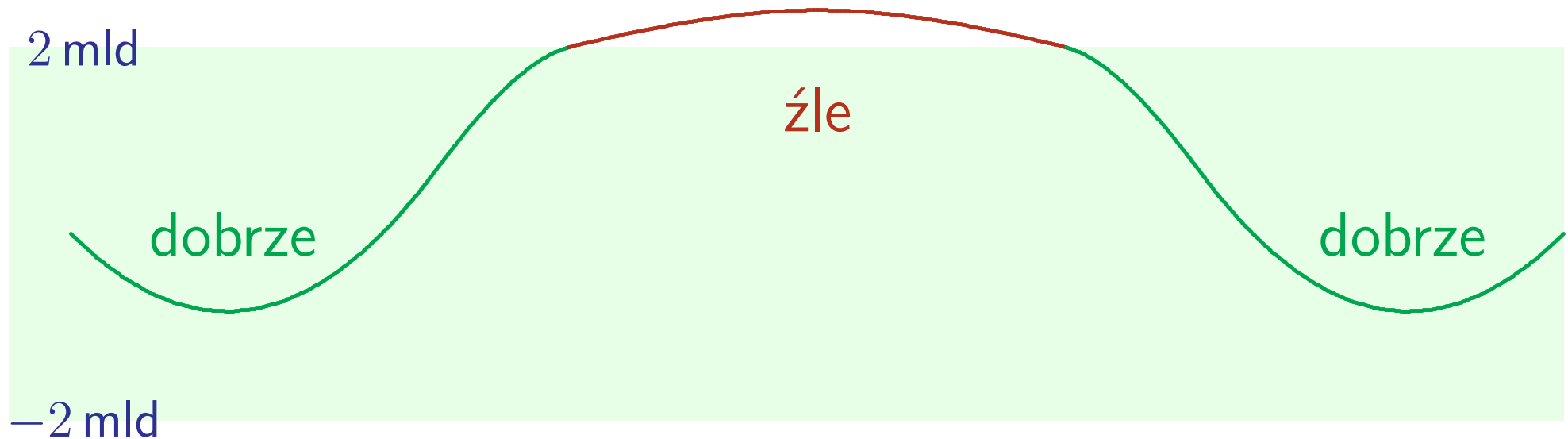
# Cudowne uleczenie wyniku



# Cudowne uleczenie wyniku



# Cudowne uleczenie wyniku

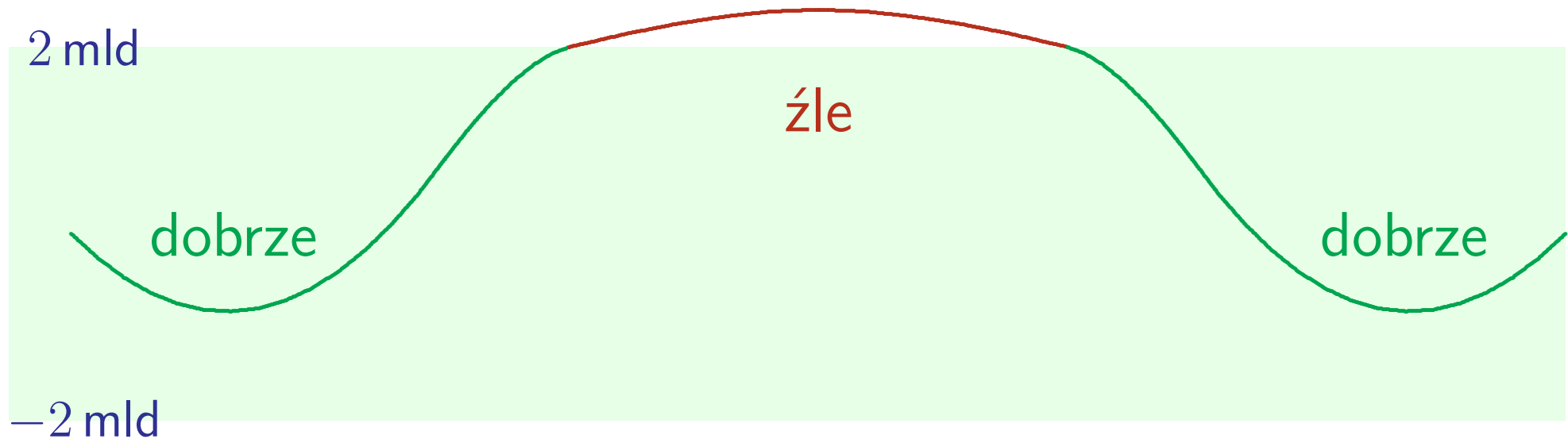


Jeśli

- w obliczeniu używamy tylko **dodawania**, **odejmowania** i **mnożenia**,
- poprawny wynik mieści się w zakresie  $[-2 \text{ mld} .. 2 \text{ mld}]$

to wynik wyliczony przez program jest **poprawny**, **nawet jeśli** obliczenie przechodzi przez liczby spoza zakresu.

# Cudowne uleczenie wyniku



Jeśli

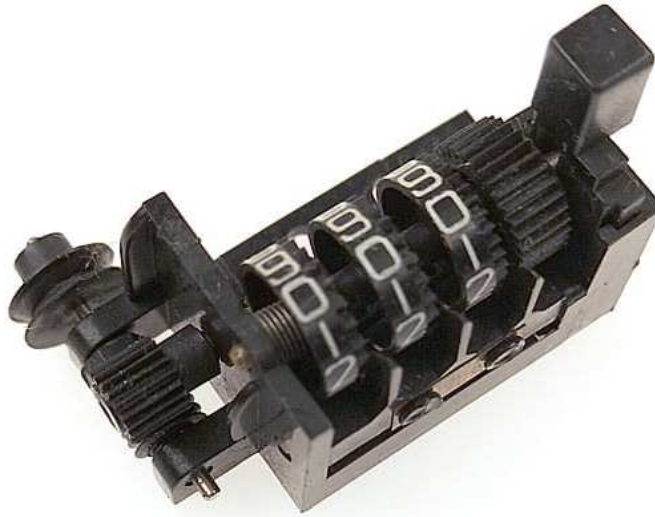
- w obliczeniu używamy tylko **dodawania**, **odejmowania** i **mnożenia**,
- poprawny wynik mieści się w zakresie  $[-2 \text{ mld} .. 2 \text{ mld}]$

to wynik wyliczony przez program jest **poprawny**, **nawet** **jeśli** obliczenie przechodzi przez liczby spoza zakresu.

## DLACZEGO?

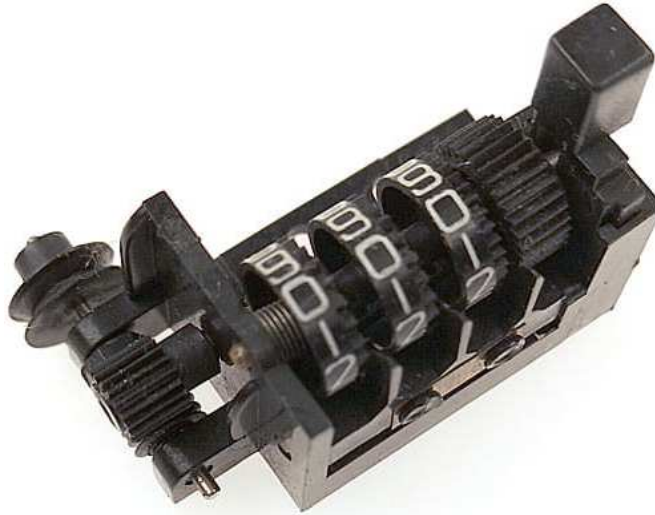
# Liczby czy liczniki z kółeczkami?...

Licznik mechaniczny:

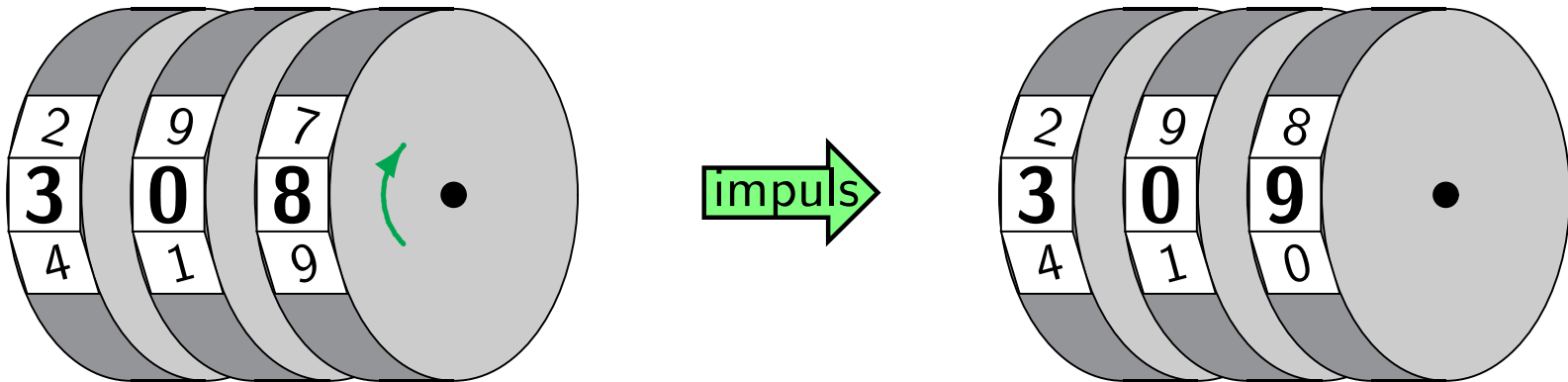


# Liczby czy liczniki z kółeczkami?...

Licznik mechaniczny:

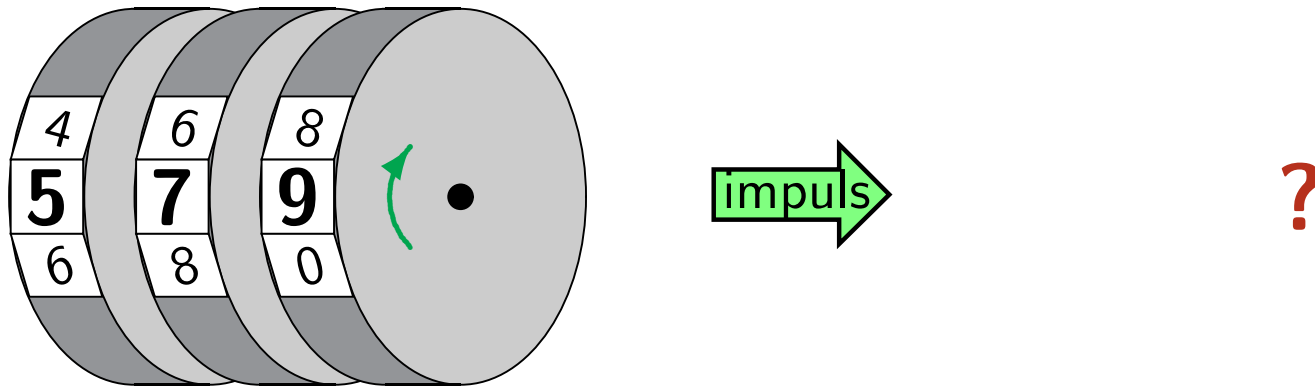


Zasada działania:



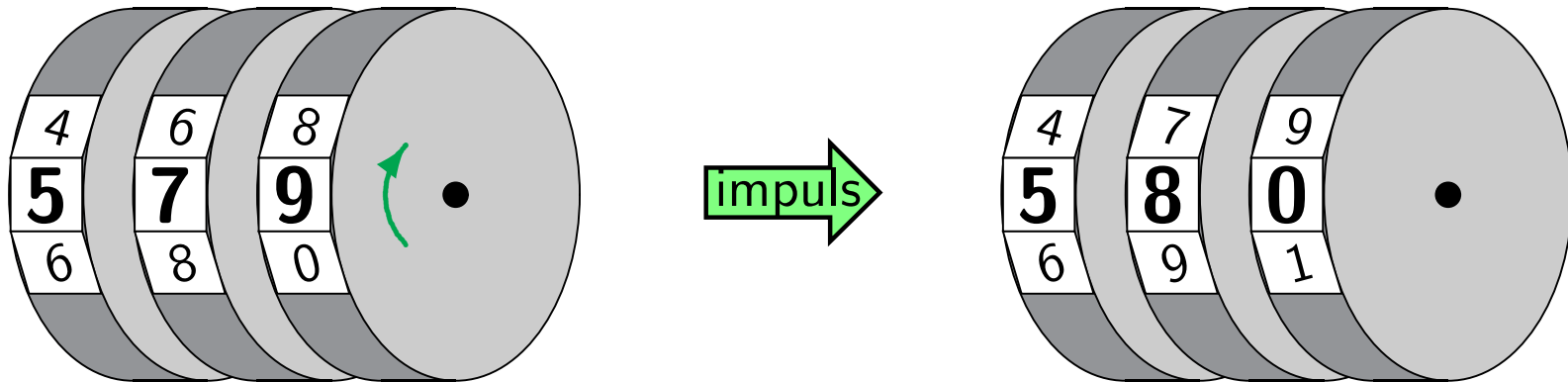
# Liczby czy liczniki z kółeczkami?...

Zasada działania:



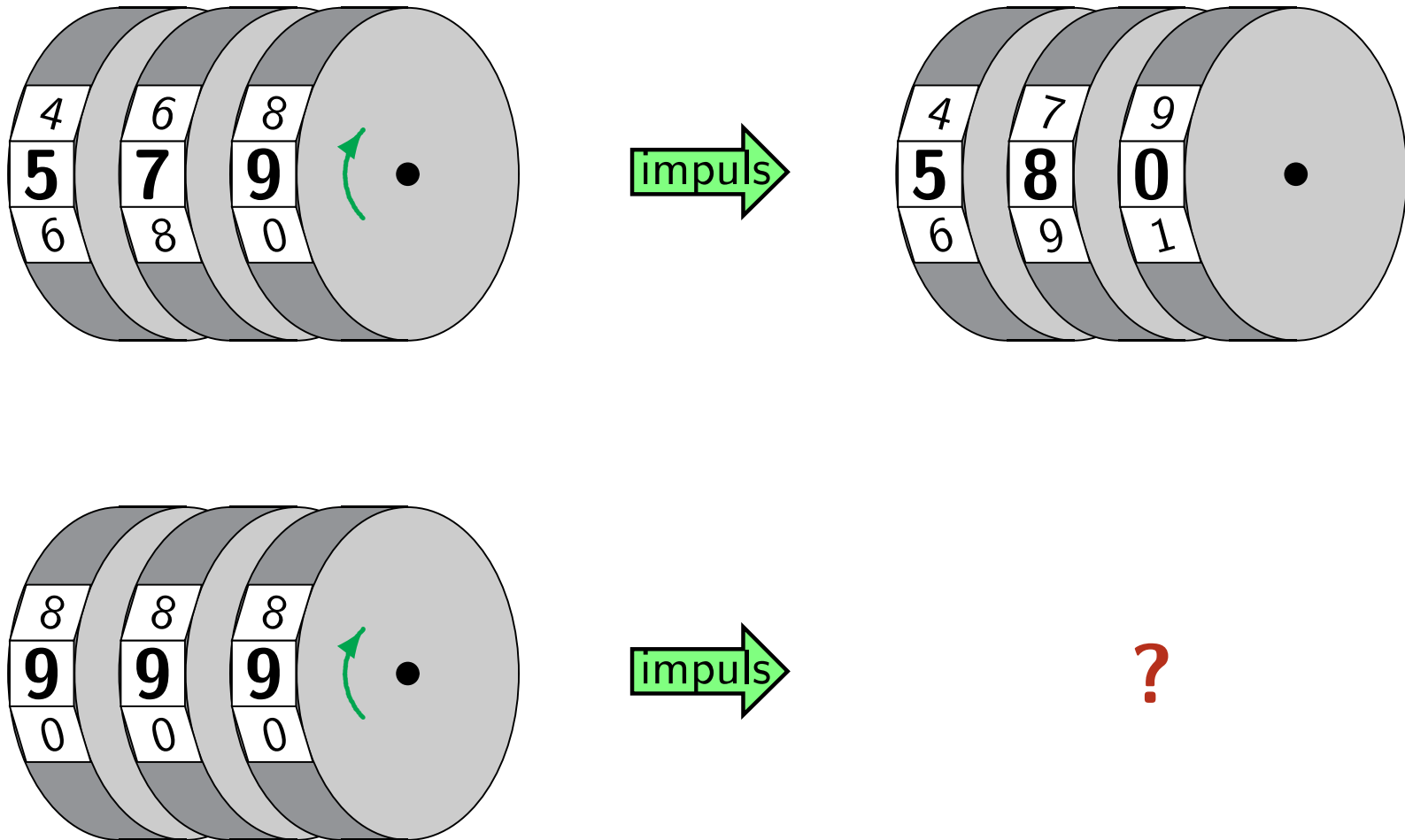
# Liczby czy liczniki z kółeczkami?...

Zasada działania:



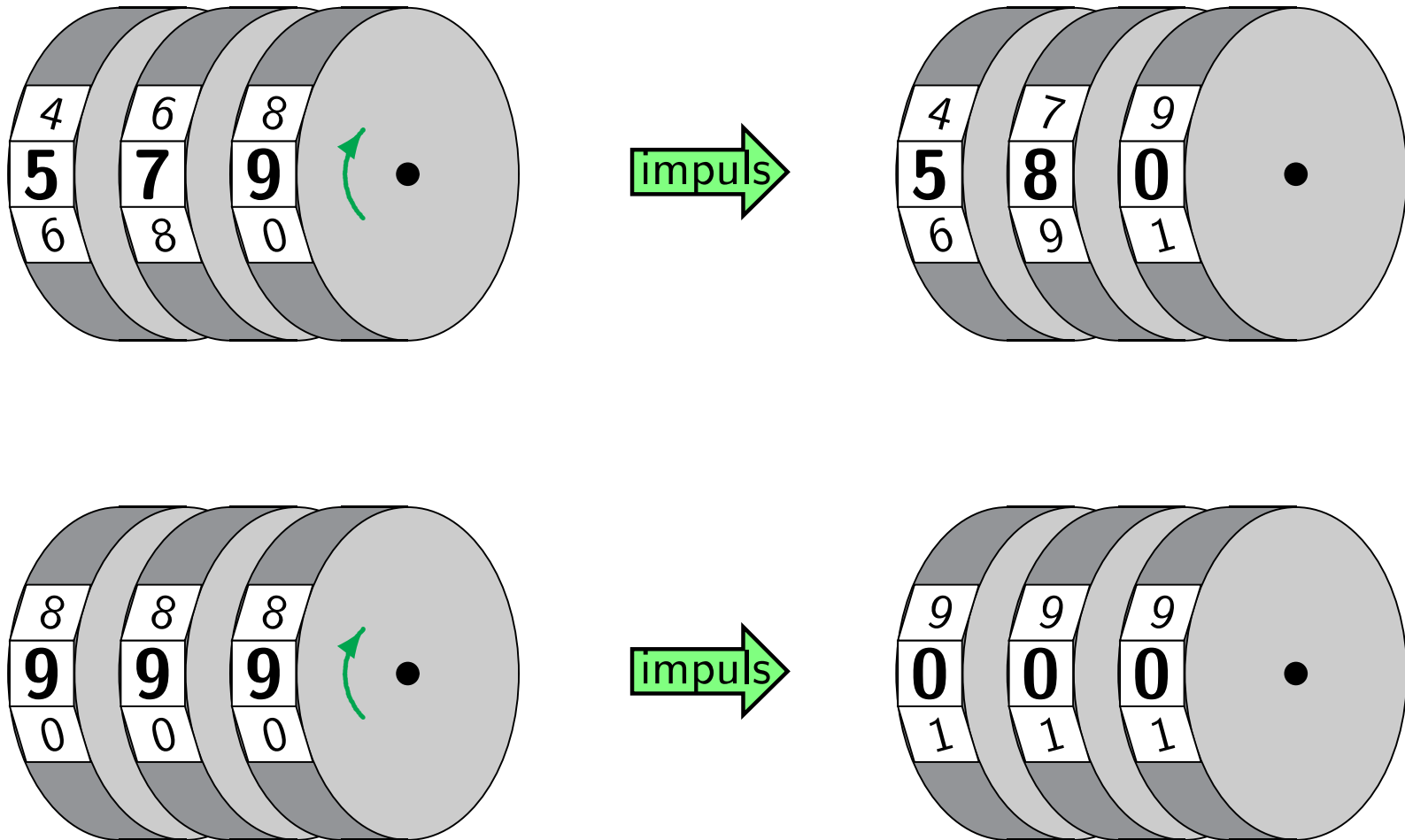
# Liczby czy liczniki z kółeczkami?...

Zasada działania:



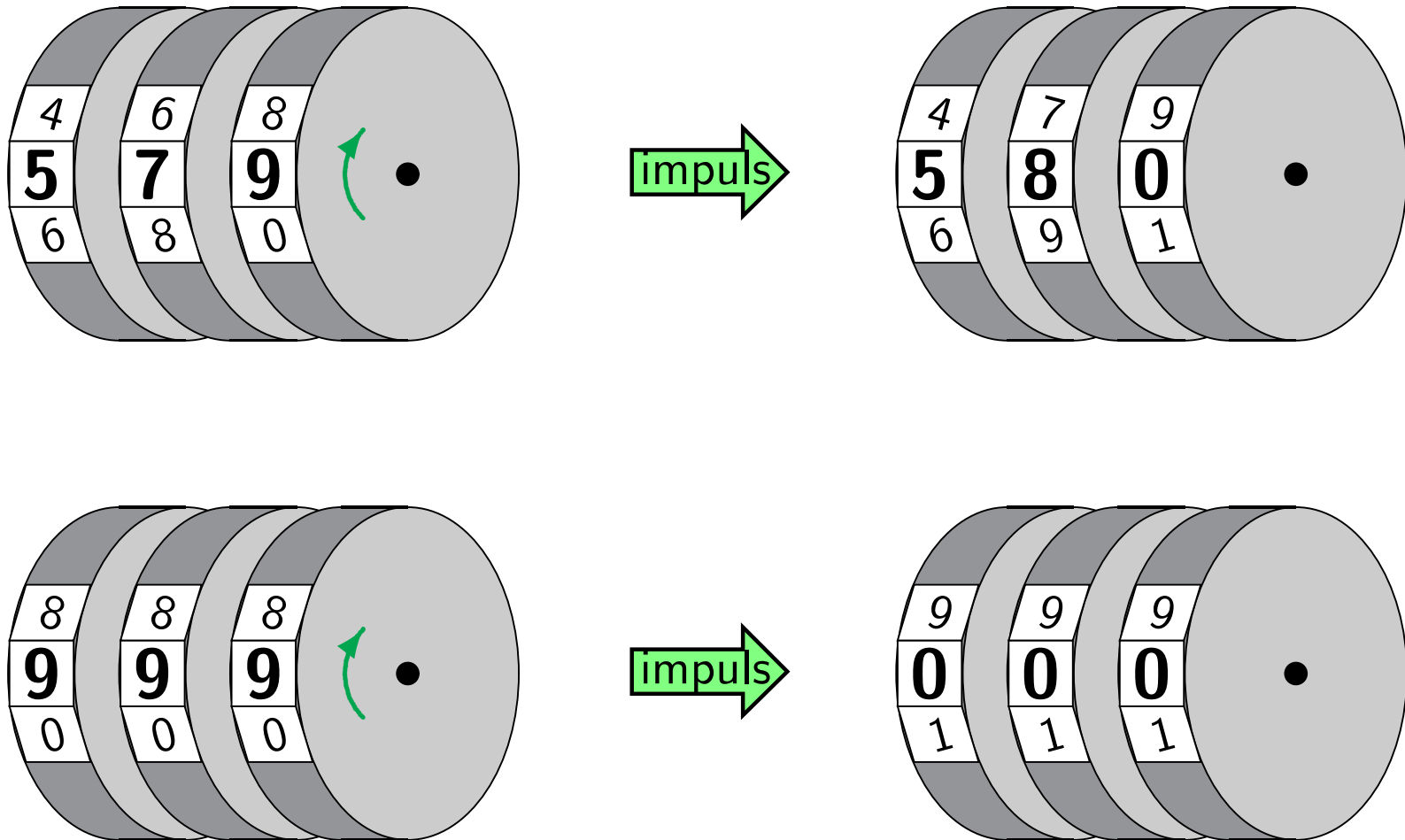
# Liczby czy liczniki z kółeczkami?...

Zasada działania:



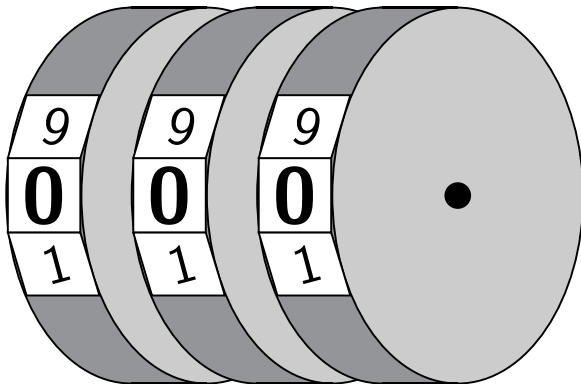
# Liczby czy liczniki z kółeczkami?...

Zasada działania:



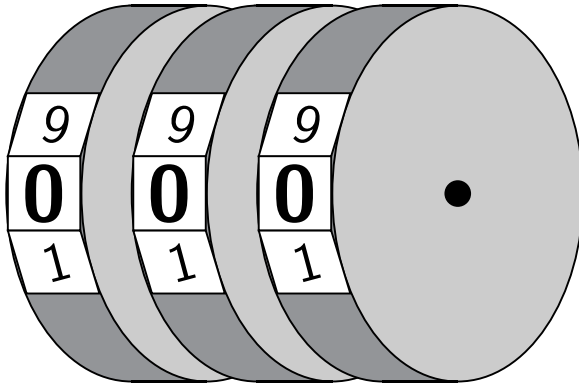
$$999 + 1 = 0 !$$

# Liczby czy liczniki z kółeczkami?...



Liczniki a liczby `int`

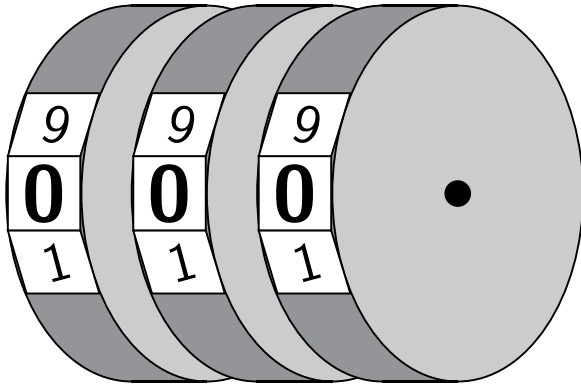
# Liczby czy liczniki z kółeczkami?...



**Liczniki a liczby `int`:**

- cyfry na kółeczku: tylko **0** lub **1**

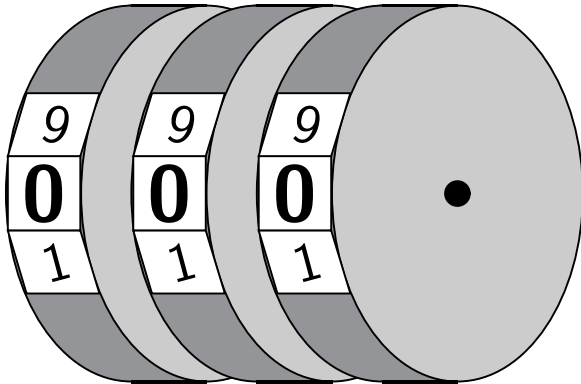
## Liczby czy liczniki z kółeczkami?...



### Liczniki a liczby `int`:

- cyfry na kółeczku: tylko **0** lub **1**;
- więcej kółeczek: 32.

# Liczby czy liczniki z kółeczkami?...



## Liczniki a liczby `int`:

- cyfry na kółeczku: tylko **0** lub **1**;
- więcej kółeczek: 32.

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
= arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

# Nie ma jak algebra...

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
= arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

# Nie ma jak algebra...

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
= arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

$$\mathbb{Z}_{2^3} = \mathbb{Z}_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

# Nie ma jak algebra...

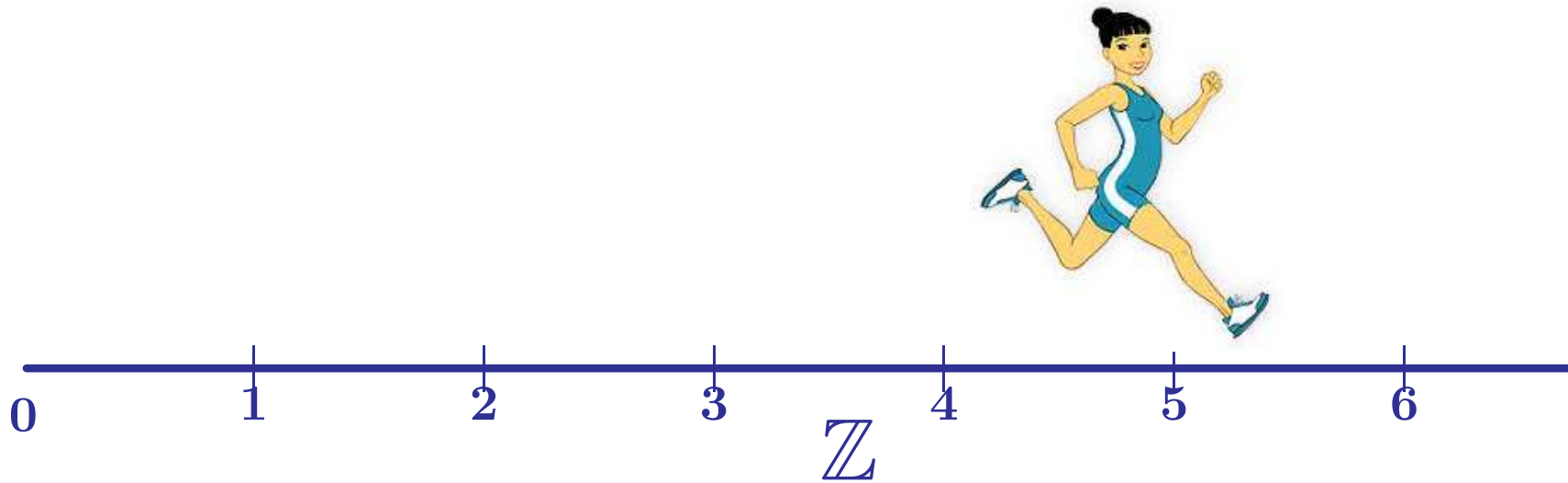
Komputerowa  $n$ -bitowa arytmetyka całkowita =  
 = arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

$$\mathbb{Z}_{2^3} = \mathbb{Z}_8 = \{0, 1, 2, 3, 4, 5, 6, 7\}$$

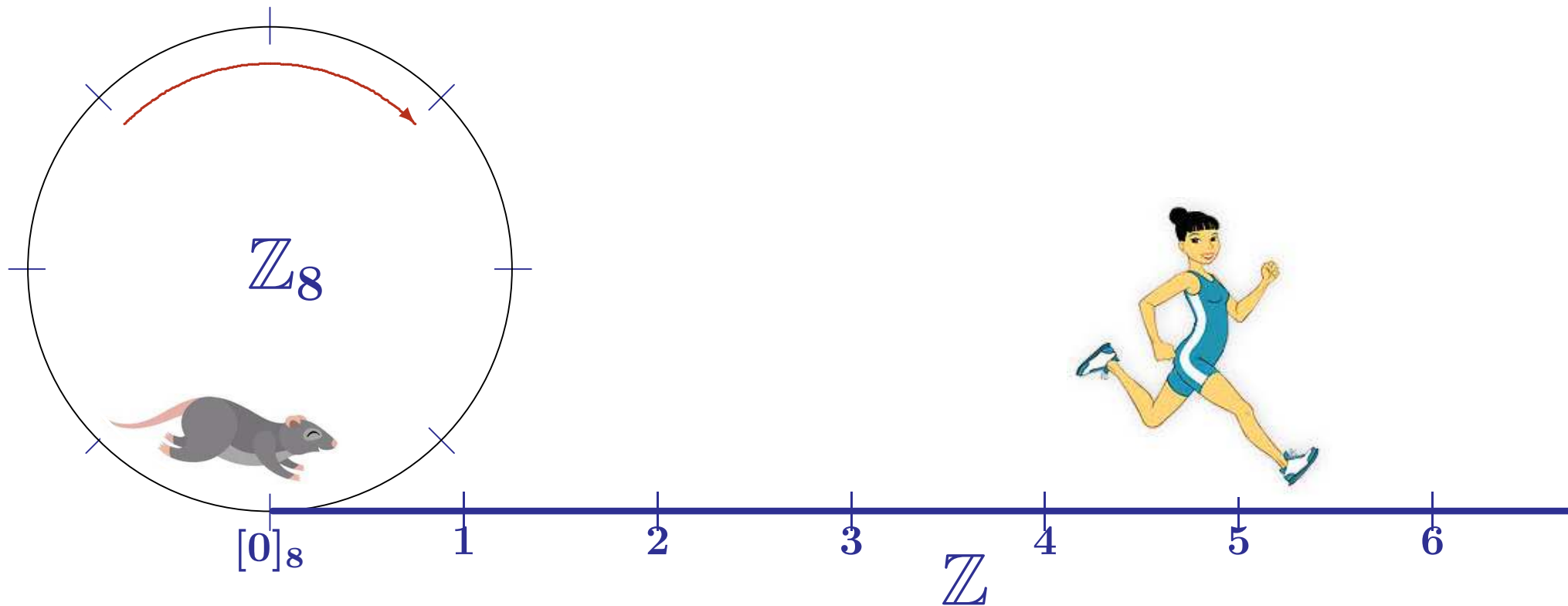
| TABLICZKA DODAWANIA |   |   |   |   |   |   |   |   |
|---------------------|---|---|---|---|---|---|---|---|
| +                   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0                   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 1                   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 |
| 2                   | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 |
| 3                   | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 |
| 4                   | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 |
| 5                   | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 |
| 6                   | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 |
| 7                   | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

| TABLICZKA MNOŻENIA |   |   |   |   |   |   |   |   |
|--------------------|---|---|---|---|---|---|---|---|
| ×                  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0                  | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1                  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 2                  | 0 | 2 | 4 | 6 | 0 | 2 | 4 | 6 |
| 3                  | 0 | 3 | 6 | 1 | 4 | 7 | 2 | 5 |
| 4                  | 0 | 4 | 0 | 4 | 0 | 4 | 0 | 4 |
| 5                  | 0 | 5 | 2 | 7 | 4 | 1 | 6 | 3 |
| 6                  | 0 | 6 | 4 | 2 | 0 | 6 | 4 | 2 |
| 7                  | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

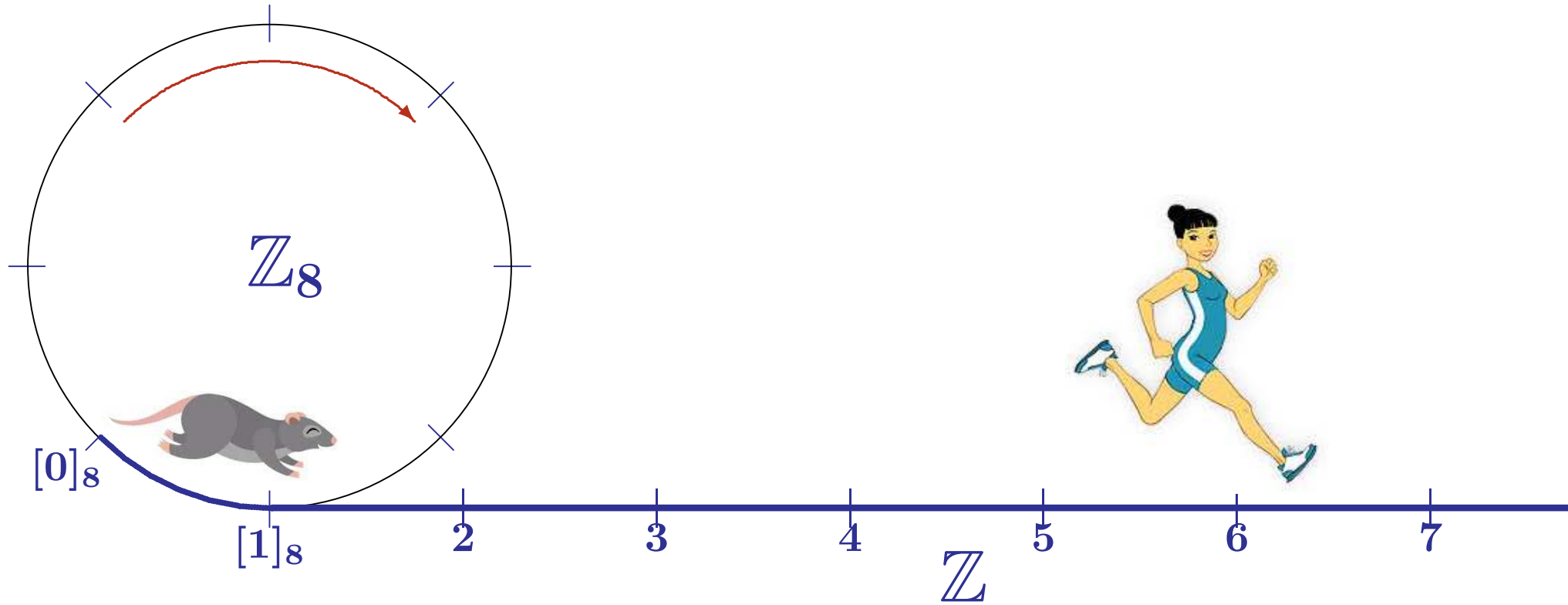
# Dookoła Wojtek



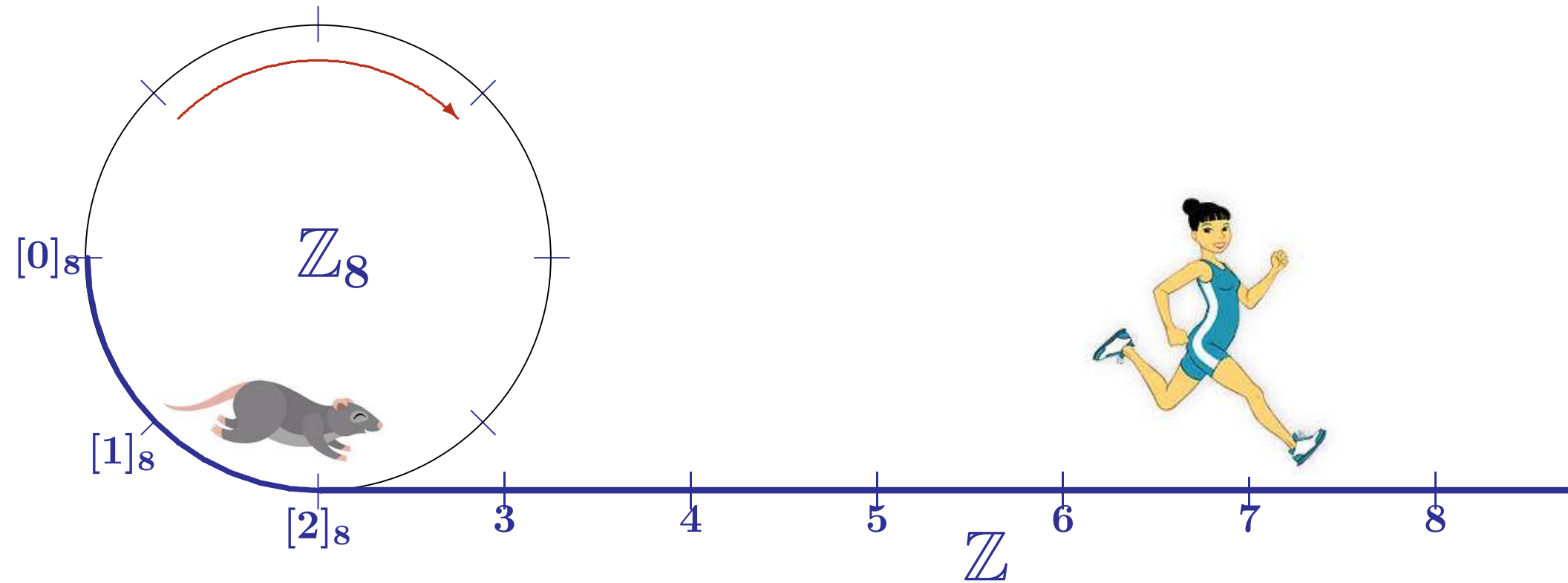
# Dookoła Wojtek



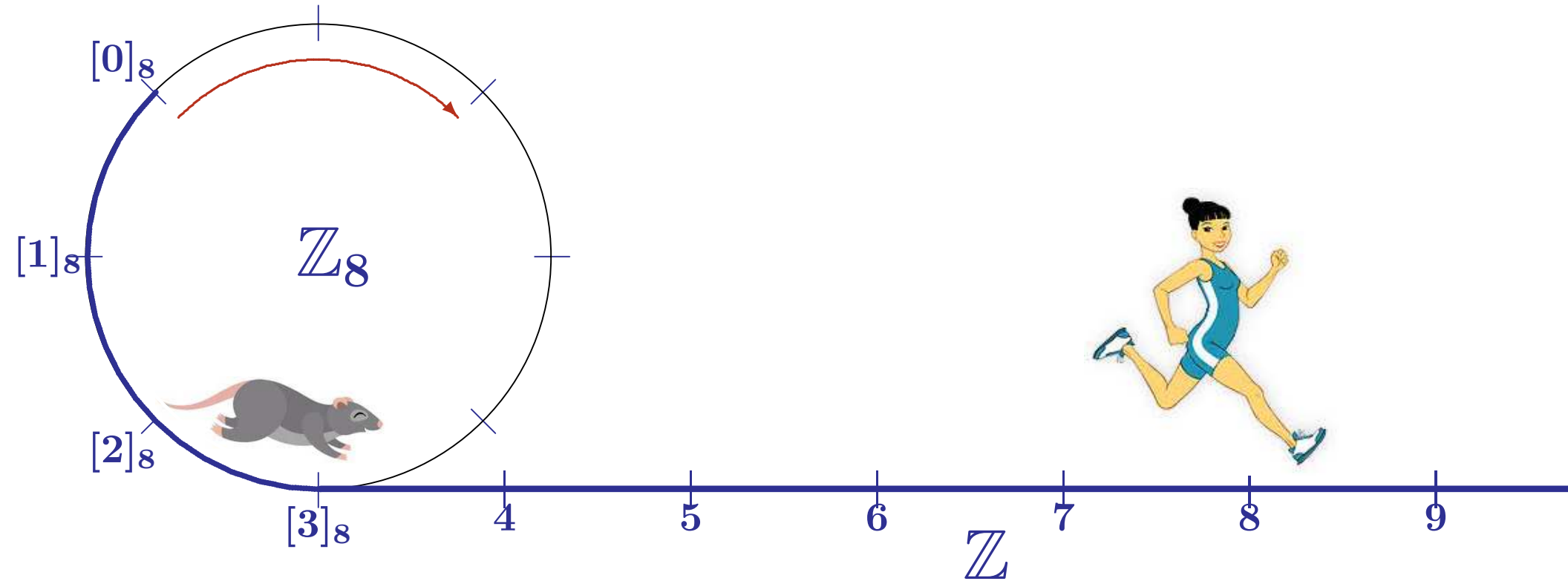
# Dookoła Wojtek



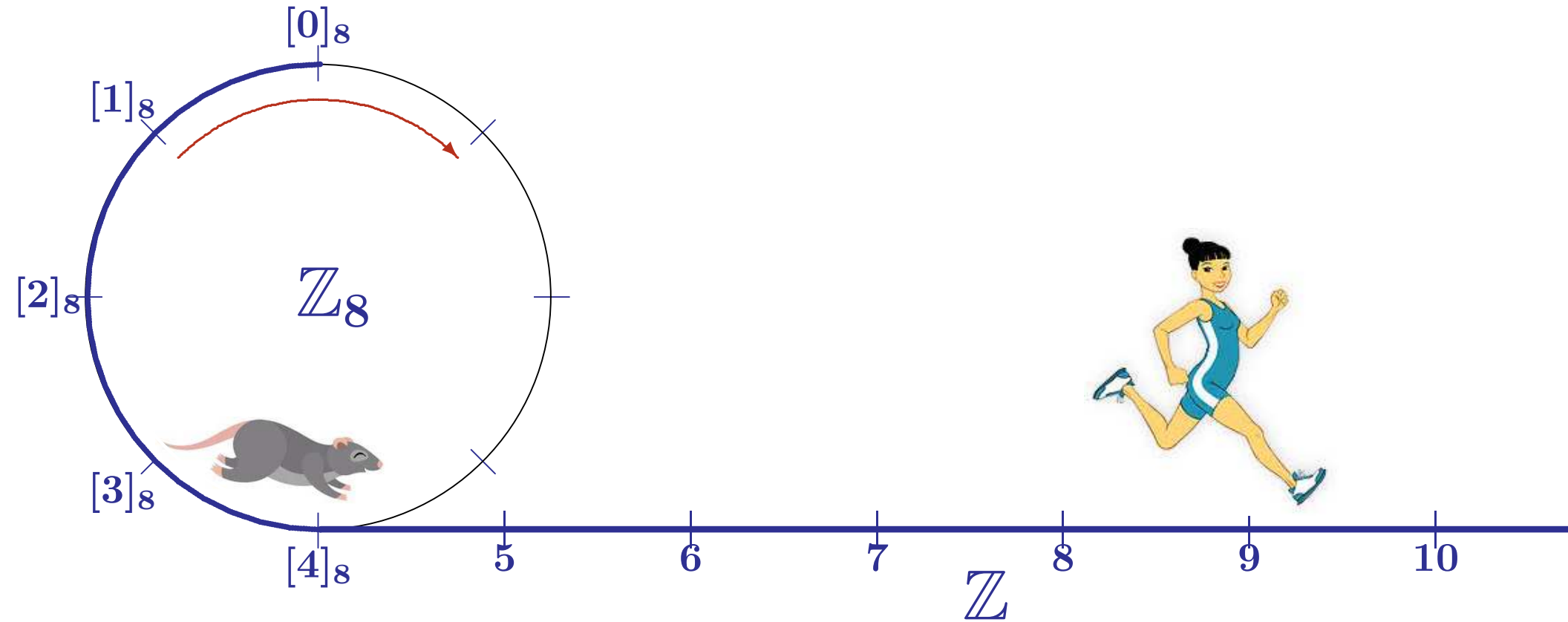
# Dookoła Wojtek



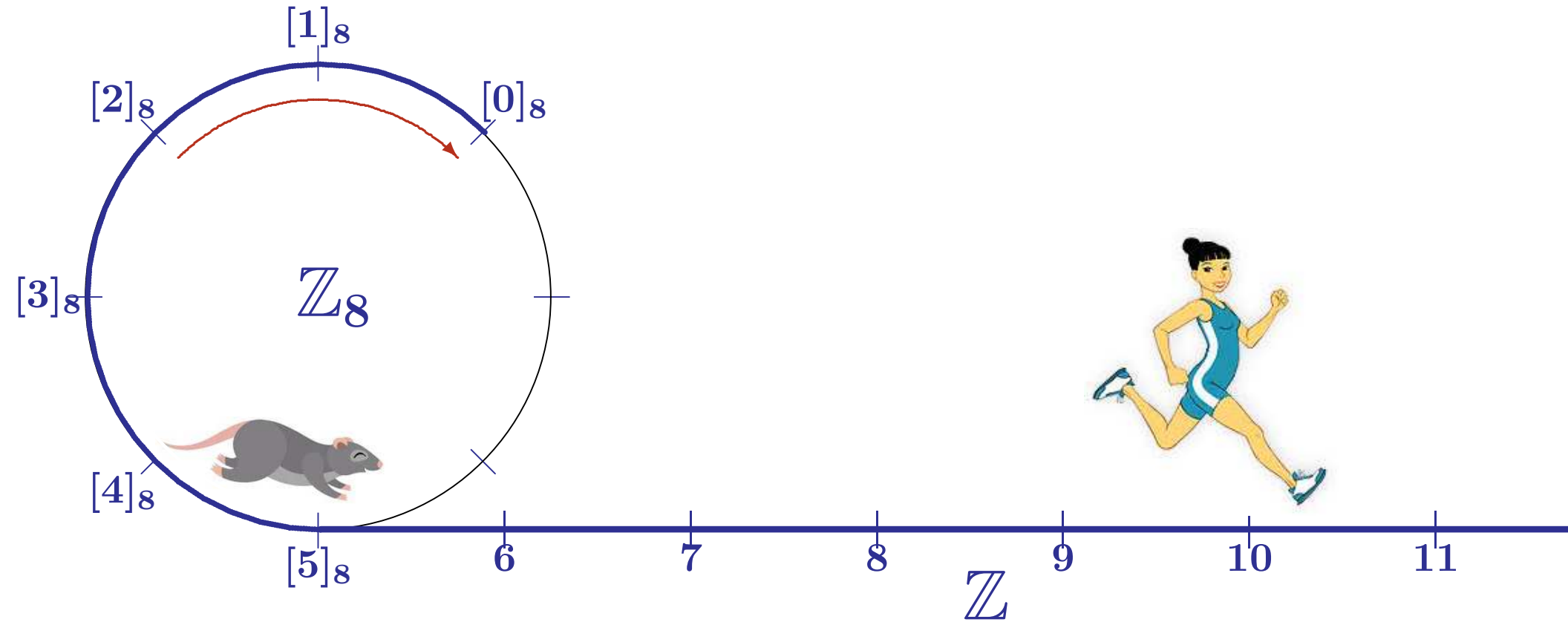
# Dokoła Wojtek



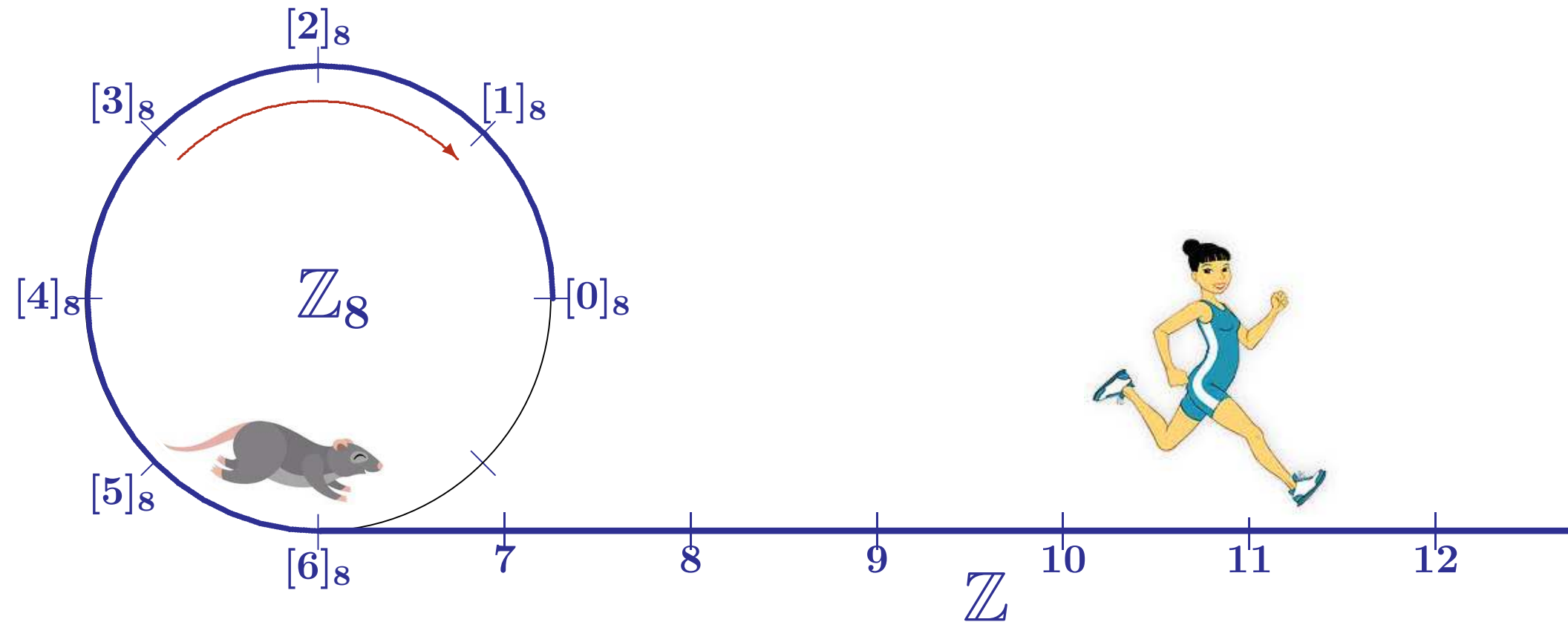
# Dookoła Wojtek



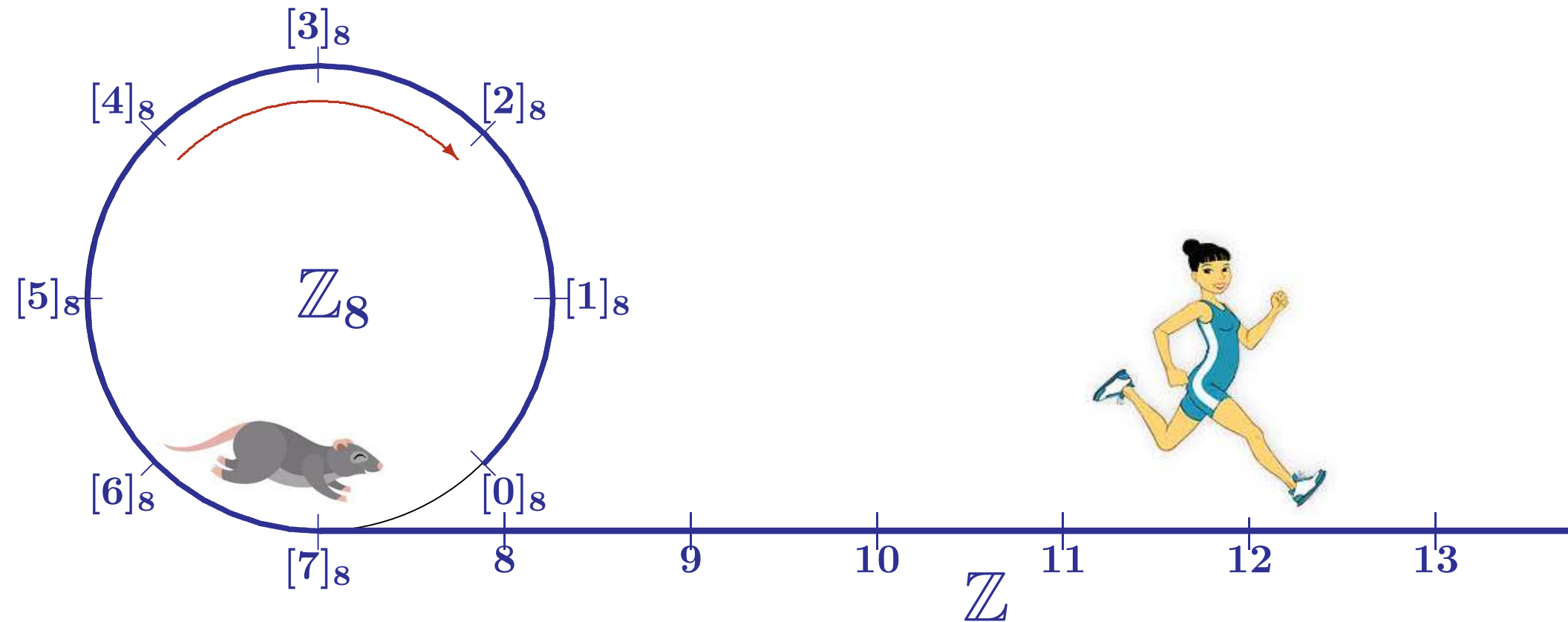
# Dookoła Wojtek



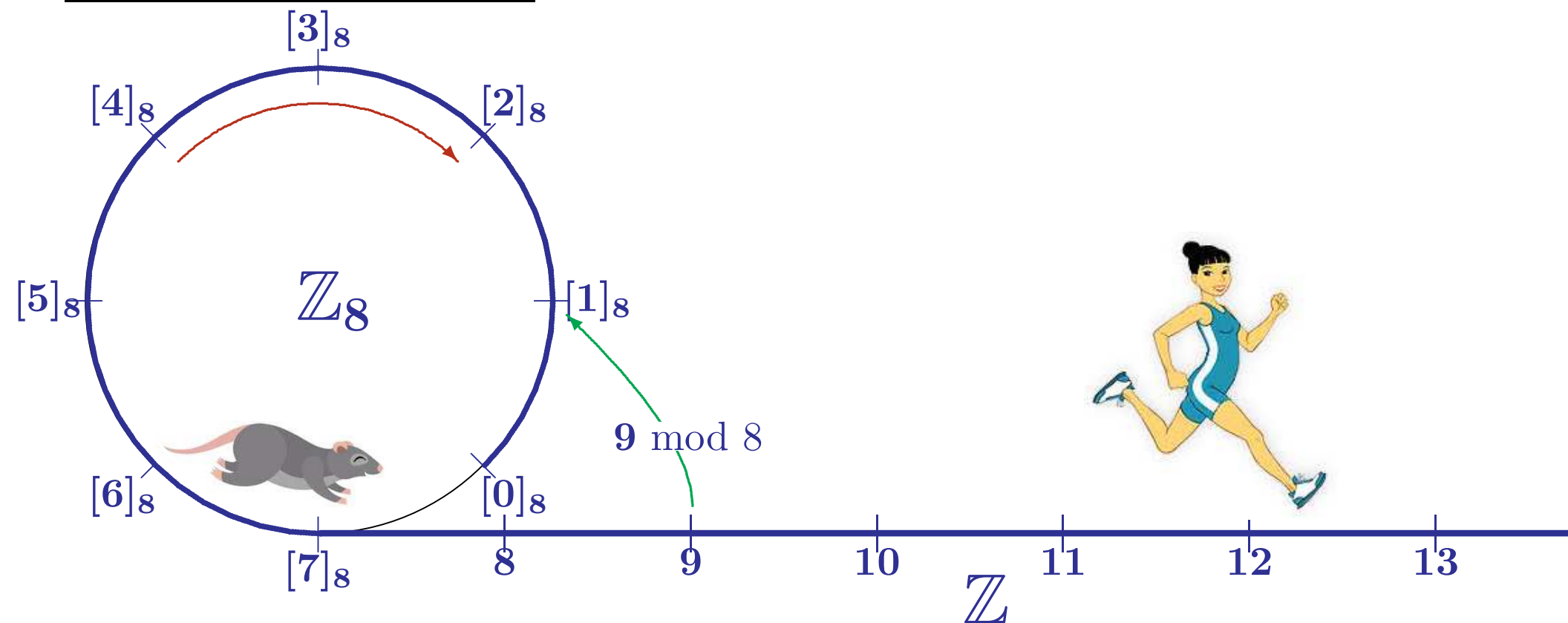
# Dookoła Wojtek



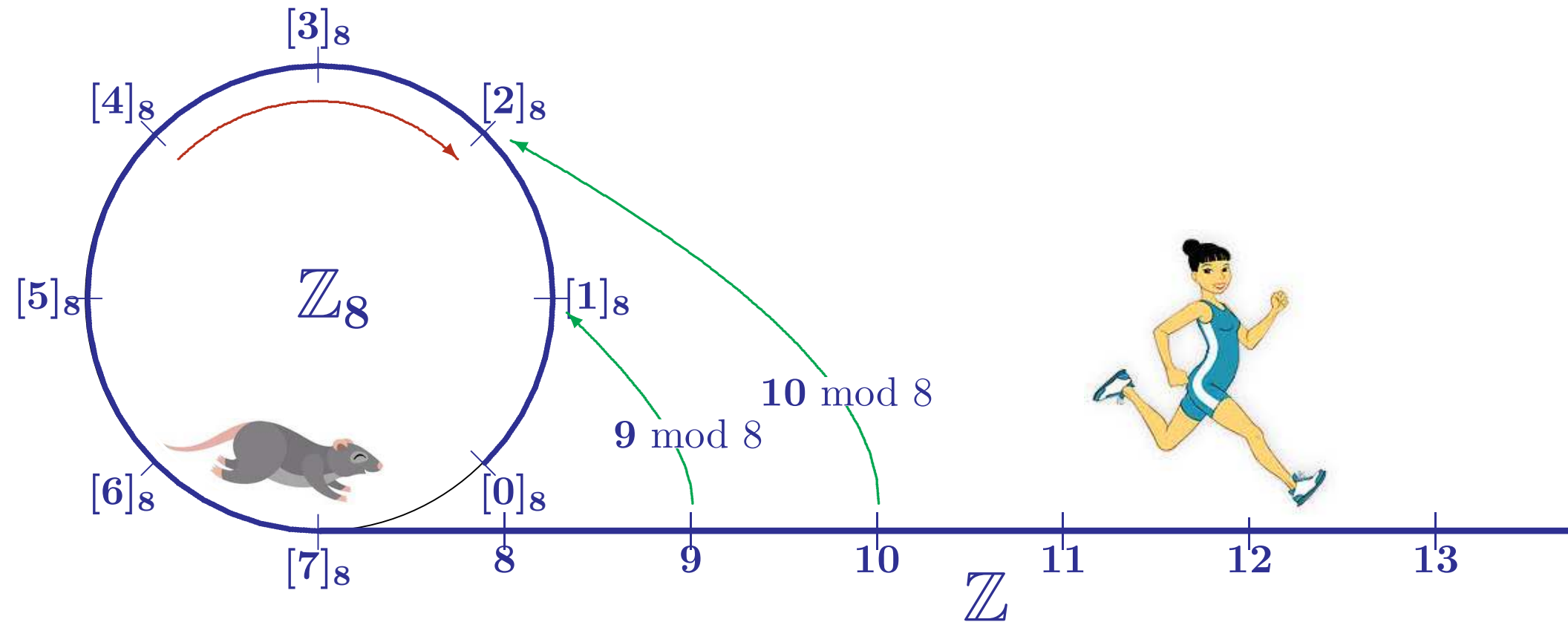
# Dookoła Wojtek



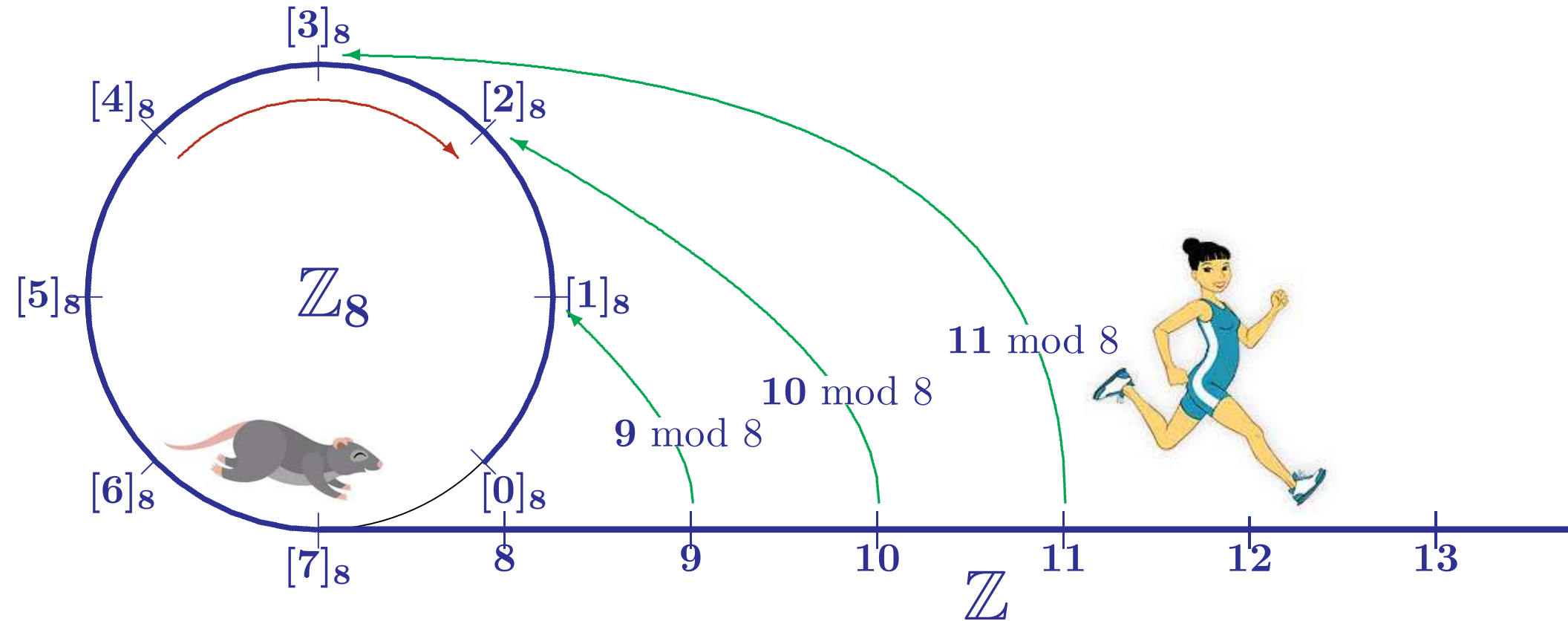
# Dookoła Wojtek



# Dookoła Wojtek

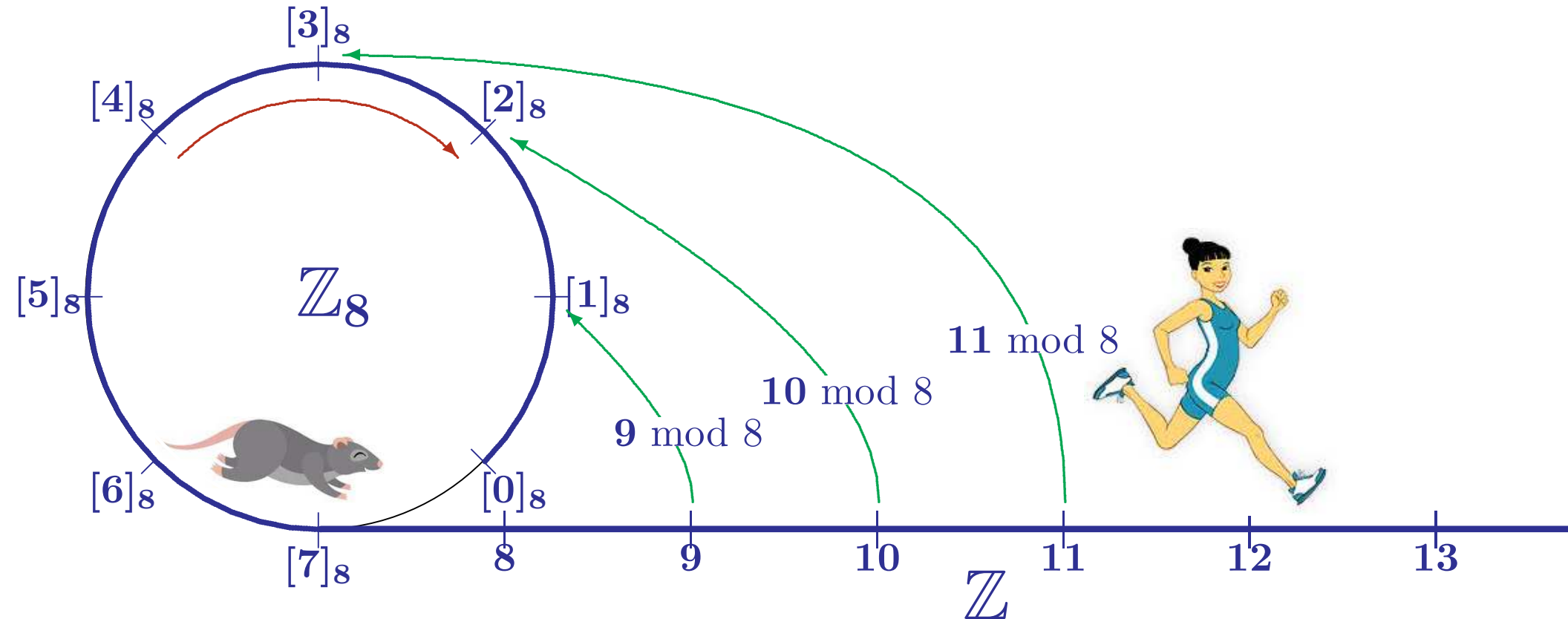


# Dookoła Wojtek



$$\dots = [-16]_8 = [-8]_8 = [0]_8 = [8]_8 = [16]_8 = \dots = \mathbf{0}$$

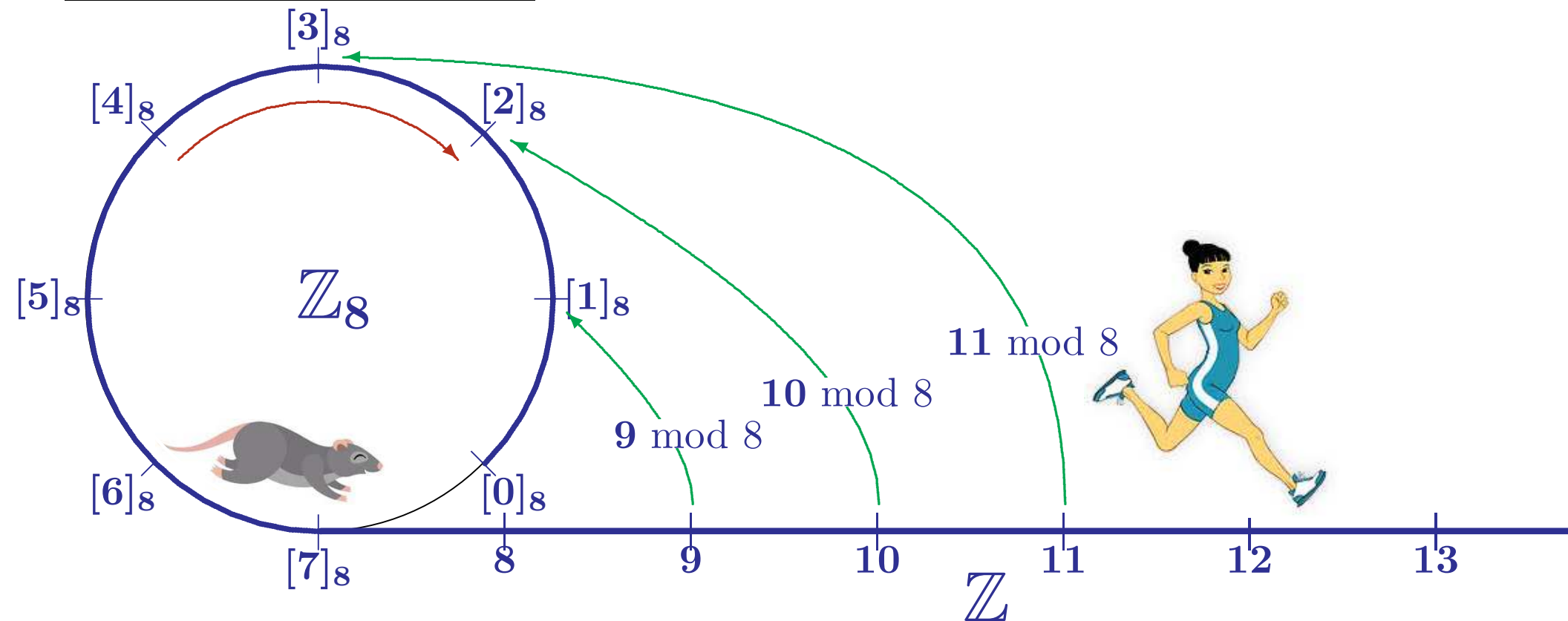
# Dookoła Wojtek



$$\dots = [-16]_8 = [-8]_8 = [0]_8 = [8]_8 = [16]_8 = \dots = \mathbf{0}$$

$$\dots = [-15]_8 = [-7]_8 = [1]_8 = [9]_8 = [17]_8 = \dots = \mathbf{1}$$

# Dookoła Wojtek

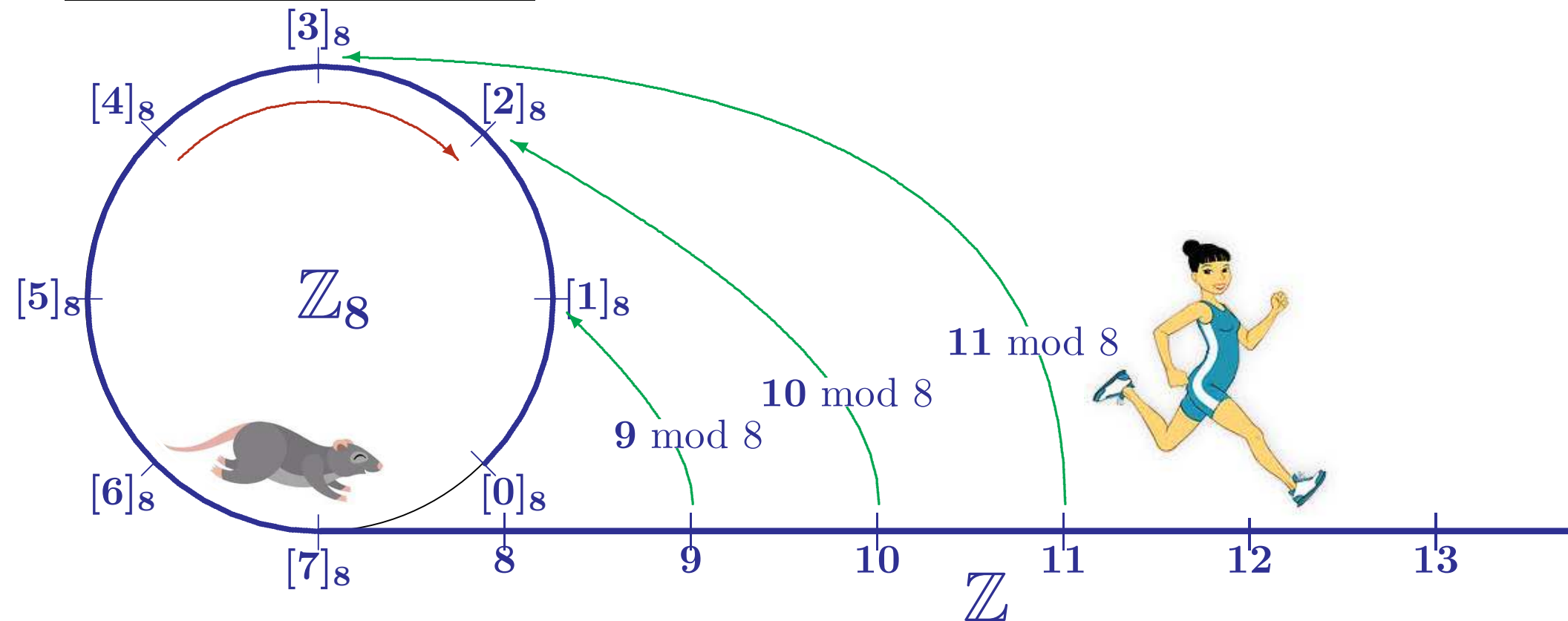


$$\dots = [-16]_8 = [-8]_8 = [0]_8 = [8]_8 = [16]_8 = \dots = \mathbf{0}$$

$$\dots = [-15]_8 = [-7]_8 = [1]_8 = [9]_8 = [17]_8 = \dots = \mathbf{1}$$

$$\dots = [-14]_8 = [-6]_8 = [2]_8 = [10]_8 = [18]_8 = \dots = \mathbf{2}$$

# Dookoła Wojtek



$$\dots = [-16]_8 = [-8]_8 = [0]_8 = [8]_8 = [16]_8 = \dots = \mathbf{0}$$

$$\dots = [-15]_8 = [-7]_8 = [1]_8 = [9]_8 = [17]_8 = \dots = \mathbf{1}$$

$$\dots = [-14]_8 = [-6]_8 = [2]_8 = [10]_8 = [18]_8 = \dots = \mathbf{2}$$

.....

# Nie ma jak algebra...

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
= arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

# Nie ma jak algebra...

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
= arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

$$\begin{cases} [\dots]_{2^n} : \mathbb{Z} \rightarrow \mathbb{Z}_{2^n} \\ [z]_{2^n} \stackrel{\text{def}}{=} z \bmod 2^n \end{cases}$$

— homomorfizm pierścieni, czyli zachowuje działania  $+$  i  $\times$

# Nie ma jak algebra...

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
= arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

$$\begin{cases} [\dots]_{2^n} : \mathbb{Z} \rightarrow \mathbb{Z}_{2^n} \\ [z]_{2^n} \stackrel{\text{def}}{=} z \bmod 2^n \end{cases}$$

— **homomorfizm pierścieni**, czyli zachowuje działania  $+$  i  $\times$

## Dlatego:

- wynik obliczenia komputerowego używającego tylko  $+$ ,  $-$  i  $\times$ , może różnić się od poprawnego o całkowitą wielokrotność modułu

# Nie ma jak algebra...

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
 = arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

$$\begin{cases} [\dots]_{2^n} : \mathbb{Z} \rightarrow \mathbb{Z}_{2^n} \\ [z]_{2^n} \stackrel{\text{def}}{=} z \bmod 2^n \end{cases}$$

— **homomorfizm pierścieni**, czyli zachowuje działania  $+$  i  $\times$

## Dlatego:

- wynik obliczenia komputerowego używającego tylko  $+$ ,  $-$  i  $\times$ , może różnić się od poprawnego o całkowitą wielokrotność modułu; dla `int` — o ok. 4 mld

# Nie ma jak algebra...

Komputerowa  $n$ -bitowa arytmetyka całkowita =  
 = arytmetyka w  $\mathbb{Z}_{2^n}$ , czyli arytmetyka reszt modulo  $2^n$

$$\begin{cases} [\dots]_{2^n} : \mathbb{Z} \rightarrow \mathbb{Z}_{2^n} \\ [z]_{2^n} \stackrel{\text{def}}{=} z \bmod 2^n \end{cases}$$

— **homomorfizm pierścieni**, czyli zachowuje działania  $+$  i  $\times$

## Dlatego:

- wynik obliczenia komputerowego używającego tylko  $+$ ,  $-$  i  $\times$ , może różnić się od poprawnego o całkowitą wielokrotność modułu; dla `int` — o ok. 4 mld;
- jeśli skądś wiemy, że poprawny wynik należy do przedziału  $[-2\text{mld}..2\text{mld}]$ , to wynik obliczenia komputerowego jest poprawny, choćby po drodze przechodziło przez bardzo duże liczby.

**A jak z dzieleniem?**

# A jak z dzieleniem?

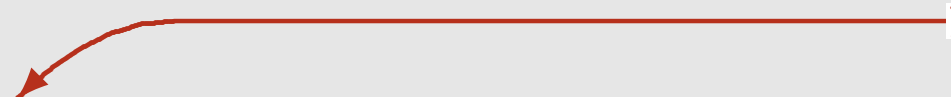
Program:

```
#include<stdio.h>
int main() {
    int p = 11100000000;
    printf("\n    (%i / 3) * 4 == %10i", p, (p/3)*4);
    printf("\n    (%i * 4) / 3 == %10i\n\n", p, (p*4)/3);
    return 0;
}
```

# A jak z dzieleniem?

Program:

```
#include<stdio.h>
int main() {
    int p = 11100000000;
    printf("\n    (%i / 3) * 4 == %10i", p, (p/3)*4);
    printf("\n    (%i * 4) / 3 == %10i\n\n", p, (p*4)/3);
    return 0;
}
```

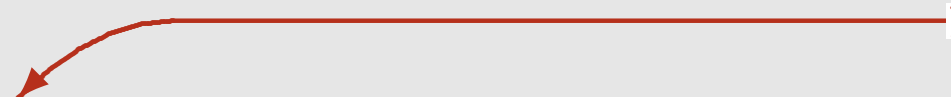


1 mld 110 mln

# A jak z dzieleniem?

Program:

```
#include<stdio.h>
int main() {
    int p = 11100000000;
    printf("\n    (%i / 3) * 4 == %10i", p, (p/3)*4);
    printf("\n    (%i * 4) / 3 == %10i\n\n", p, (p*4)/3);
    return 0;
}
```



1 mld 110 mln

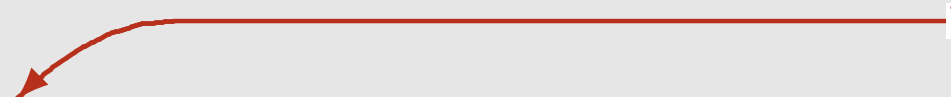
wyświetla:

```
(11100000000 / 3) * 4 == 14800000000
(11100000000 * 4) / 3 ==    48344234
```

# A jak z dzieleniem?

Program:

```
#include<stdio.h>
int main() {
    int p = 11100000000;
    printf("\n    (%i / 3) * 4 == %10i", p, (p/3)*4);
    printf("\n    (%i * 4) / 3 == %10i\n\n", p, (p*4)/3);
    return 0;
}
```



1 mld 110 mln

wyświetla:

```
(11100000000 / 3) * 4 == 14800000000
(11100000000 * 4) / 3 ==    48344234
```

**Mnożenie jest nieprzemienne?!**

# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone.

## Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną*

# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity

# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

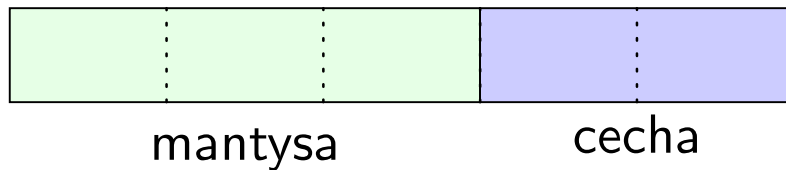
- licznik jest całkowity,
- mianownik jest potęgą 2.

# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity,
- mianownik jest potęgą 2.

Zapis komputerowy *zmiennopozycyjny*:

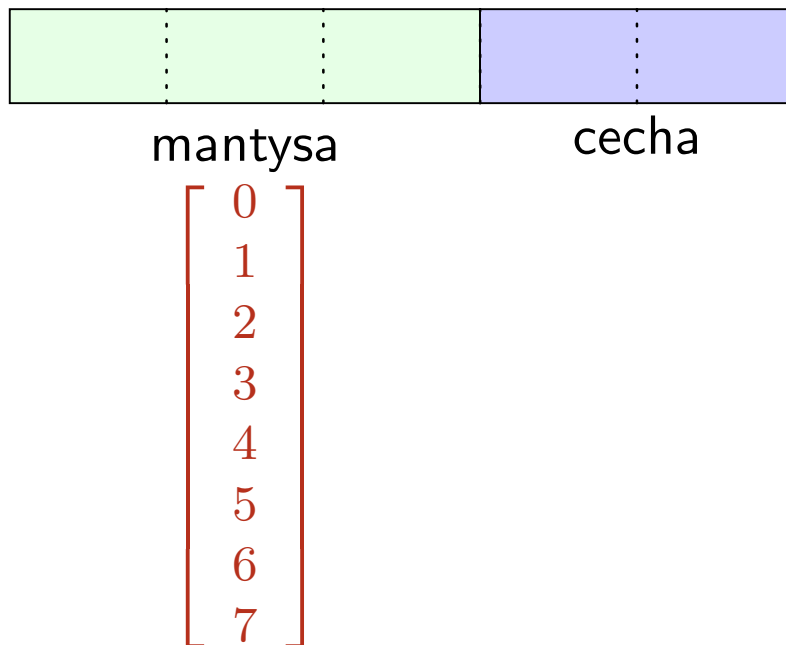


# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity,
- mianownik jest potęgą 2.

Zapis komputerowy *zmiennopozycyjny*:

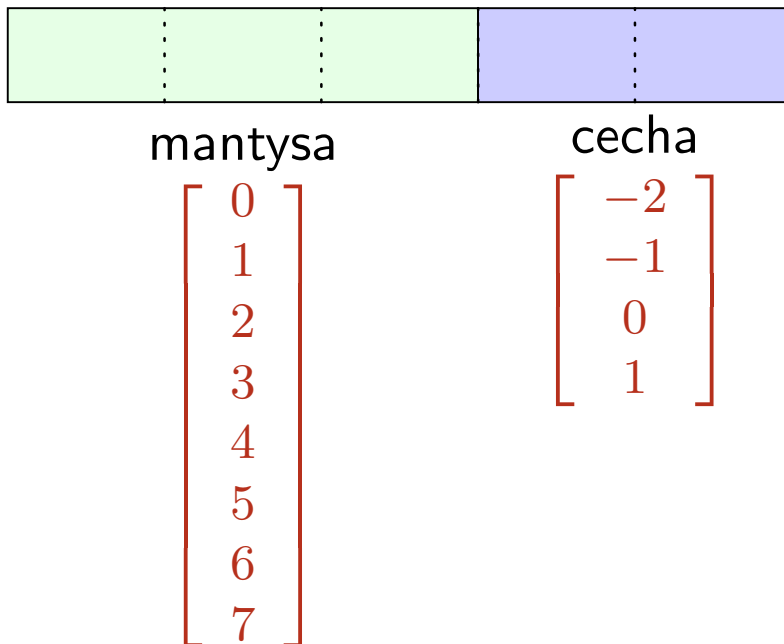


# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity,
- mianownik jest potęgą 2.

Zapis komputerowy *zmiennopozycyjny*:

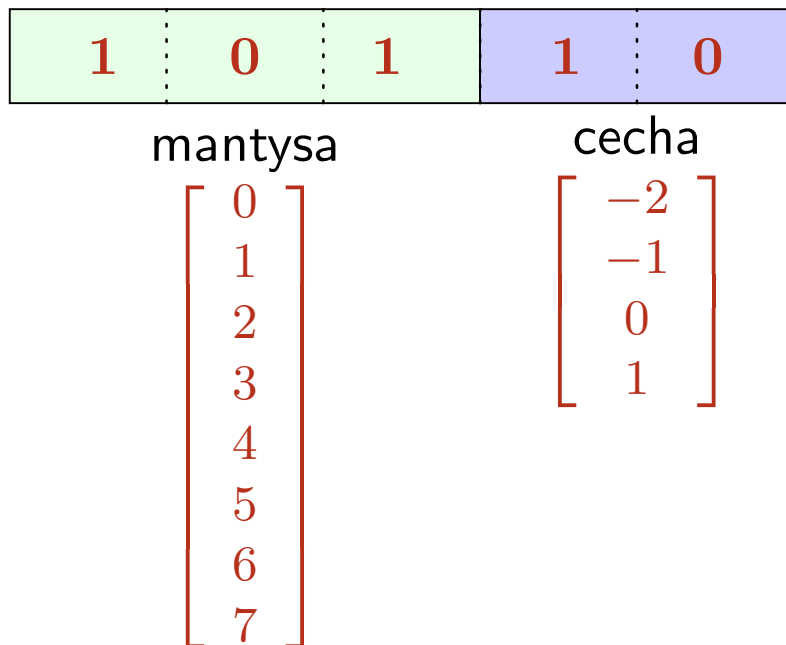


# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity,
- mianownik jest potęgą 2.

Zapis komputerowy *zmiennopozycyjny*:



# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity,
- mianownik jest potęgą 2.

Zapis komputerowy *zmiennopozycyjny*:

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|

$$= (1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}) \cdot 2^{-1 \cdot 2^1 + 0 \cdot 2^0}$$

mantysa

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix}$$

cecha

$$\begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity,
- mianownik jest potęgą 2.

Zapis komputerowy *zmiennopozycyjny*:

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|

$$= (1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}) \cdot 2^{-1 \cdot 2^1 + 0 \cdot 2^0}$$

$$= 1\frac{1}{4} \cdot 2^{-2}$$

mantysa

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix}$$

cecha

$$\begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

# Nierzeczywistość liczb rzeczywistych

**Wszystkie** obliczenia na liczbach rzeczywistych są zaokrąglone. Liczba rzeczywista jest reprezentowana przez bliską jej liczbę *dwójkowo wymierną* — czyli taki ułamek, którego

- licznik jest całkowity,
- mianownik jest potęgą 2.

Zapis komputerowy *zmiennopozycyjny*:

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 0 | 1 | 1 | 0 |
|---|---|---|---|---|

$$= (1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2}) \cdot 2^{-1 \cdot 2^1 + 0 \cdot 2^0}$$

$$= 1\frac{1}{4} \cdot 2^{-2} = \frac{5}{16} = 0.3125$$

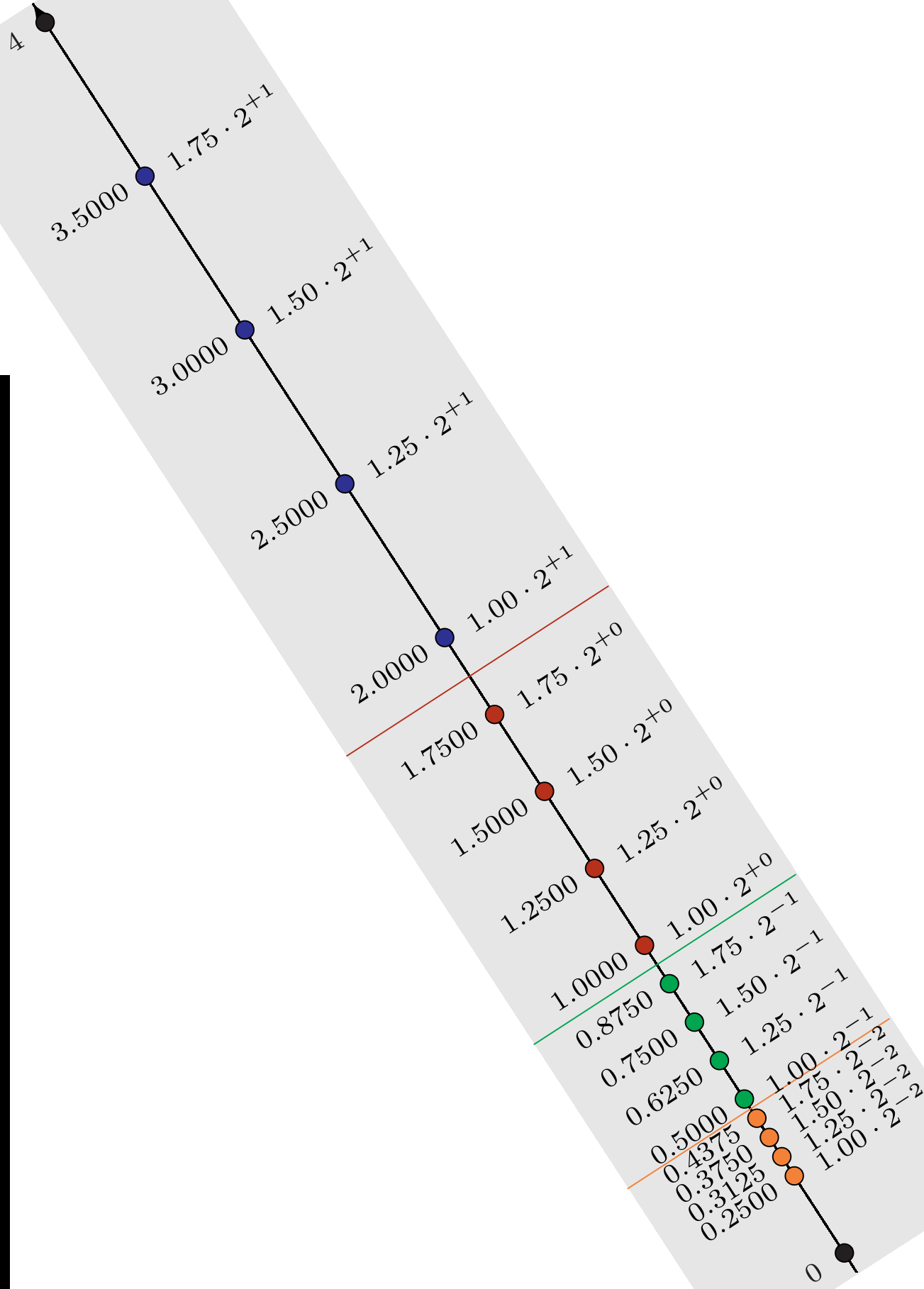
mantysa

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{bmatrix}$$

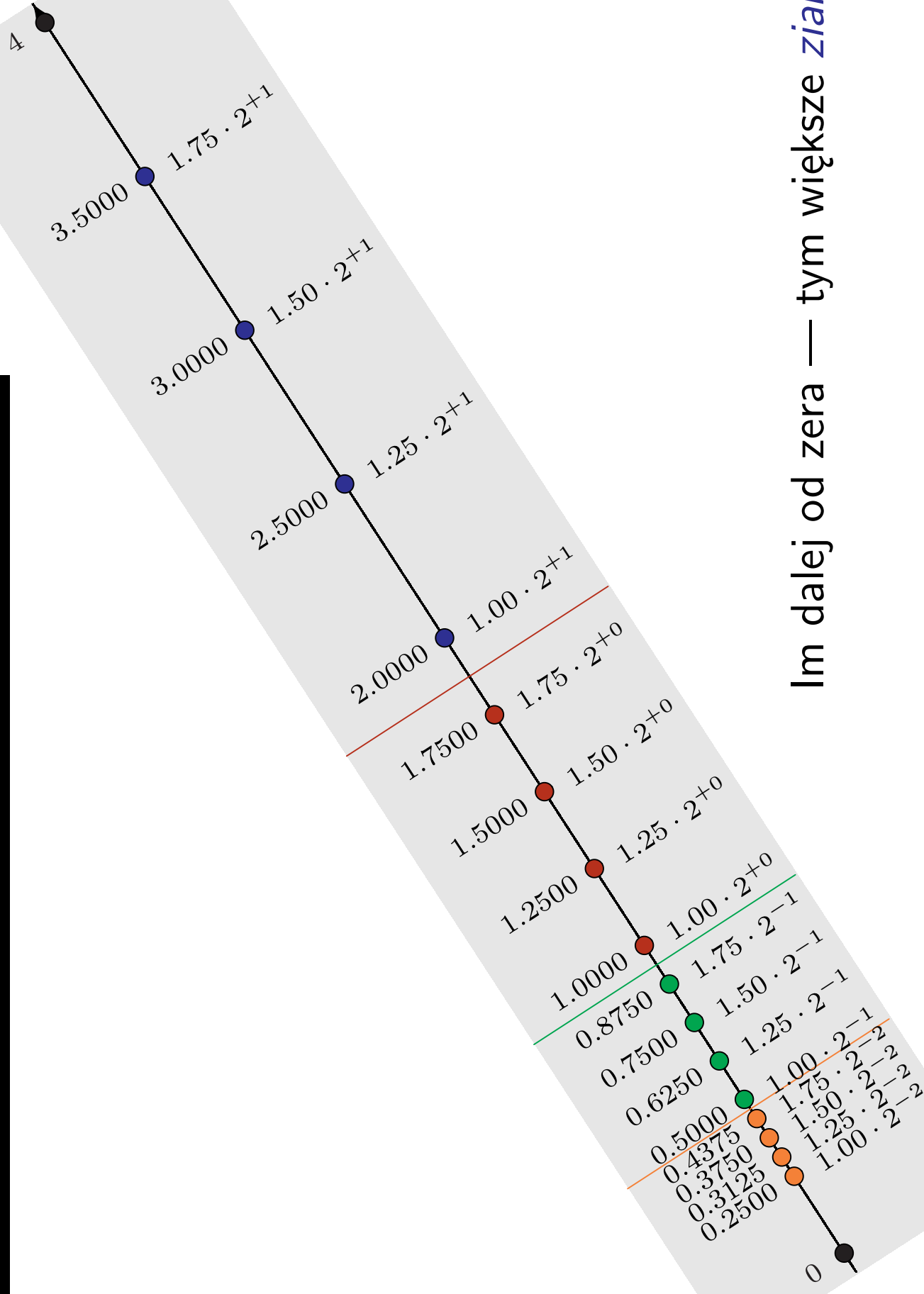
cecha

$$\begin{bmatrix} -2 \\ -1 \\ 0 \\ 1 \end{bmatrix}$$

# Nierzeczywistość liczb rzeczywistych



# Nierzeczywistość liczb rzeczywistych



Im dalej od zera — tym większe *ziarno*.

# Opłakane skutki zaokrąglania

- Test `if (x==y) ...` nie ma sensu.

# Opłakane skutki zaokrąglania

- Test `if (x==y) ...` nie ma sensu.  
... chcesz uniknąć 100 przykrości, to zapomnij o równości ...

# Opłakane skutki zaokrąglania

- Test `if (x==y) ...` nie ma sensu.  
... chcesz uniknąć 100 przykrości, to zapomnij o równości ...
- Może się zdarzyć, że  $x + 1 = x$ .

# Opłakane skutki zaokrąglania

- Test `if (x==y) ...` nie ma sensu.  
... chcesz uniknąć 100 przykrości, to zapomnij o równości ...
- Może się zdarzyć, że  $x + 1 = x$ .
- Dodawanie nie jest łączne:  $(x + 1) + 1 \neq x + (1 + 1)$ .

# Opłakane skutki zaokrąglania

- Test `if (x==y) ...` nie ma sensu.  
... chcesz uniknąć 100 przykrości, to zapomnij o równości ...
- Może się zdarzyć, że  $x + 1 = x$ .
- Dodawanie nie jest łączne:  $(x + 1) + 1 \neq x + (1 + 1)$ .
- Wzory skróconego mnożenia nie działają:  $(x + 1)^2 < x^2 + 2x + 1$ .

# Marzenie a spełnienie

# Marzenie a spełnienie

Rozwój techniki spełnia marzenia — ale niedokładnie.

## **Marzenie a spełnienie**

Rozwój techniki spełnia marzenia — ale niedokładnie.

Skutek bywa większy niż marzenie, ale trudny do opanowania.

## Marzenie a spełnienie

Rozwój techniki spełnia marzenia — ale niedokładnie.

Skutek bywa większy niż marzenie, ale trudny do opanowania.

matematyka jest  
abstrakcyjna i idealna

## Marzenie a spełnienie

Rozwój techniki spełnia marzenia — ale niedokładnie.

Skutek bywa większy niż marzenie, ale trudny do opanowania.

matematyka jest  
abstrakcyjna i idealna

komputer jest  
konkretny i techniczny

## Marzenie a spełnienie

Rozwój techniki spełnia marzenia — ale niedokładnie.

Skutek bywa większy niż marzenie, ale trudny do opanowania.

|                                           |   |                              |
|-------------------------------------------|---|------------------------------|
| matematyka jest<br>abstrakcyjna i idealna | } | — więc czasem się pogryzą... |
| komputer jest<br>konkretny i techniczny   |   |                              |

## Marzenie a spełnienie

Rozwój techniki spełnia marzenia — ale niedokładnie.

Skutek bywa większy niż marzenie, ale trudny do opanowania.

|                                           |   |                              |
|-------------------------------------------|---|------------------------------|
| matematyka jest<br>abstrakcyjna i idealna | } | — więc czasem się pogryzą... |
| komputer jest<br>konkretny i techniczny   |   |                              |

Żeby poruszać się po jezdni lub w powietrzu, **trzeba znać** niebezpieczeństwa.

## Marzenie a spełnienie

Rozwój techniki spełnia marzenia — ale niedokładnie.

Skutek bywa większy niż marzenie, ale trudny do opanowania.

|                                           |   |                              |
|-------------------------------------------|---|------------------------------|
| matematyka jest<br>abstrakcyjna i idealna | } | — więc czasem się pogryzą... |
| komputer jest<br>konkretny i techniczny   |   |                              |

Żeby poruszać się po jezdni lub w powietrzu, **trzeba znać** niebezpieczeństwa.

Żeby posługiwać się komputerem, **trzeba wiedzieć**, kiedy mu ufać i do jakiego stopnia.

## Wniosek

Zalecam użytkownikom komputerów

**OGRANICZONE ZAUFANIE**

## Wniosek

Zalecam użytkownikom komputerów

**OGRANICZONE ZAUFANIE**

ale w żadnym razie nie zalecam

**BEZGRANICZNEJ NIEUFNOŚCI**