O abstrakcyjnym myśleniu

Jacek Cichoń
jacek.cichon@pwr.edu.pl

Politechnika Wrocławska
Wydział Podstawowych Problemów Techniki
Katedra Informatyki

Jelenia Góra, 10 października, 2019

Operacje matematyczne

Dodawanie

- $2 + 3 = 5$; $\frac{1}{2} + \frac{1}{3} = \frac{5}{6}$; $4.21 + 2.15 = 6.36$
- suma dwóch liczb całkowitych jest liczbą całkowitą;
- suma dwóch liczb wymiernych jest liczbą wymierną;
- suma dwóch liczb rzeczywistych jest liczbą rzeczywistą.

Mnożenie

- $2 \cdot 3 = 6$; $\frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$; $4.21 \cdot 2.15 = 9.0515$
- iloczyn dwóch liczb naturalnych jest liczbą naturalną;
- iloczyn dwóch liczb wymiernych jest liczbą wymierną;
- iloczyn dwóch liczb rzeczywistych jest liczbą rzeczywistą.

Właściwości ułatwiające obliczenia - I

- $(2+3)+5 = 2+(3+5)$
- $(\frac{1}{2}+\frac{1}{3})+\frac{1}{5} = \frac{1}{2}+(\frac{1}{3}+\frac{1}{5})$
- $(2.34 \cdot 3.42) \cdot 5.12 = 2.34 \cdot (3.42 \cdot 5.12)$

Definition (łączność)

Działanie $\oplus$ jest łączne jeśli

$$x \oplus (y \oplus z) = (x \oplus y) \oplus z$$

dla dowolnych x, y, z (z ustalonej dziedziny).

Wniosek

Dodawanie i mnożenie w liczbach całkowitych, wymiernych, rzeczywistych jest łączne.

Właściwości ułatwiające obliczenia - II

- $2 + 3 = 2 + 3$
- $\frac{3}{5} + \frac{3}{7} = \frac{3}{7} + \frac{3}{5}$
- $4.123 \cdot 7.213 = 7.213 \cdot 4.123$

Definition (przemienność)

Działanie $\star$ jest przemienne jeśli

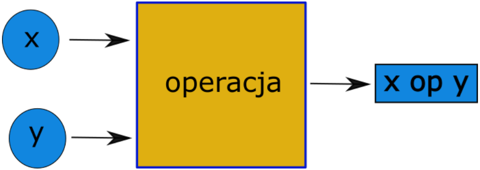
$$x \star y = y \star x$$

dla dowolnych x, y (z ustalonej dziedziny).

Wniosek

Dodawanie i mnożenie w liczbach całkowitych, wymiernych, rzeczywistych jest przemienne.

Operacje arytmetyczne



Przykłady operacji:

- przykład: $+$ (dodawanie)
- przykład: $\cdot$ (mnożenie)

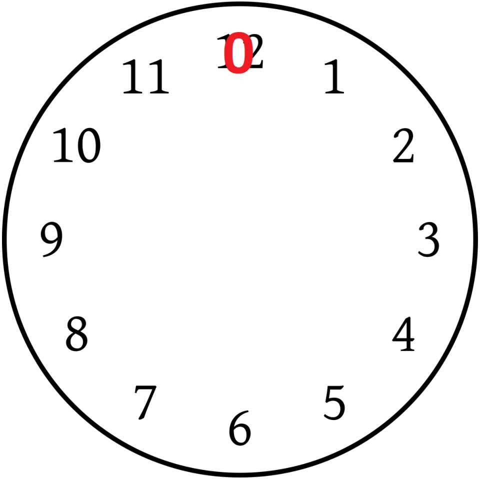
Inny przykład (w liczbach naturalnych)

$$x \# y = x^y$$

1. $2\#3 = 8 \neq 9 = 3\#2$
2. przykład: $\cdot$ (mnożenie)

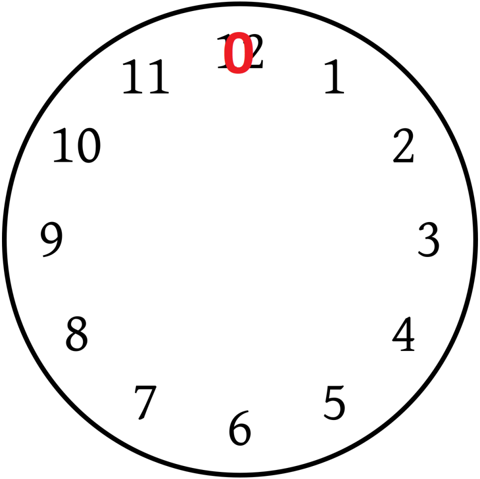
$$2\#(3\#2) = 512 \neq 64 = (2\#3)\#2$$

Arytmetyka zegarka



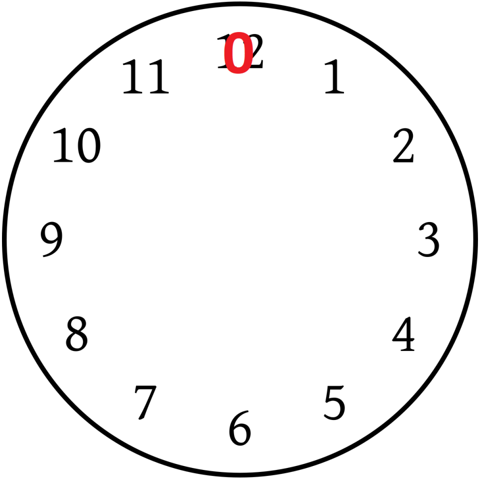
- $6 + 2 = 8$
- $6 + 3 = 9$
- $6 + 4 = 10$
- $6 + 5 = 11$

Arytmetyka zegarka



- $6 + 2 = 8$
- $6 + 3 = 9$
- $6 + 4 = 10$
- $6 + 5 = 11$
- $6 + 6 = 0$
- $6 + 7 = 1$

Arytmetyka zegarka



- $6 + 2 = 8$
- $6 + 3 = 9$
- $6 + 4 = 10$
- $6 + 5 = 11$
- $6 + 6 = 0$
- $6 + 7 = 1$
- $8 + 8 = 4$
- $8 + 9 = 5$
- $8 + 10 = 6$

Arytmetyka zegarka: II

Jak dodajemy w arytmetyce zegarka ?

$$\mathbf{8} + \mathbf{9} = 17 = 1 \cdot 12 + 5 = \mathbf{5}$$

$$\mathbf{9} + \mathbf{11} = 20 = 1 \cdot 12 + 8 = \mathbf{8}$$

Algorytm

1. oblicz $x + y$
2. znajdź takie k i r, że $x + y = k \cdot 12 + r$, że $0 \leq r < 12$
3. połóż $x \oplus_{12} y = r$

Arytmetyka zegarka: III

Daleki skok: mnożymy

$$\mathbf{5} \cdot \mathbf{3} = 15 = 1 \cdot 12 + 3 = \mathbf{3}$$

$$\mathbf{5} \cdot \mathbf{5} = 25 = 2 \cdot 12 + 1 = \mathbf{1}$$

Algorytm

1. oblicz $x \cdot y$
2. znajdź takie k i r, że $x \cdot y = k \cdot 12 + r$, że $0 \leq r < 12$
3. połóż $x \odot_{12} y = r$

Arytmetyka zegarka: IV

Pierścień $\mathbb{Z}_{12}$

$$\mathbb{Z}_{12} = (\{0, 1, 2, \ldots, 11\}, \oplus_{12}, \odot_{12})$$

Podstawowe własności

1. $\oplus_{12}$, $\odot_{12}$ są łączne i przemienne
2. $(x \oplus_{12} y) \odot_{12} z = (x \odot_{12} z) \oplus_{12} (y \odot_{12} z)$ (rozdzielność $\oplus_{12}$ względem $\oplus_{12}$)
3. $x \oplus_{12} 0 = 0 \oplus_{12} x = x$ (element 0 jest elementem neutralnym dodawania)
4. $x \odot_{12} 1 = 1 \odot_{12} x = x$ (element 1 jest elementem neutralnym mnożenia)
5. ...

Co się nam przydarzyło ?

Przed wprowadzeniem pojęcia działania

$$4 + 3 = 7, \quad 5 \cdot 7 = 35$$

$$23655432 \cdot 44219764 = 1046037620358048$$

Po wprowadzeniu pojęcia działania

$$\mathbb{Z}_2, \quad ,\mathbb{Z}_3, \quad \mathbb{Z}_4, \quad \mathbb{Z}_5 \quad \ldots \quad \mathbb{Z}_{2333211234}, \quad \ldots$$

Dział matematyki, który zajmuje się tego typu strukturami nazywa się

Algebrą Abstrakcyjną

Zasada Abstrakcji

Relacja równoważności

Dla par $x, y \in \Omega$ mamy określoną zależność $\equiv$ o następujących własnościach:

- $x \equiv x$ dla dowolnego $x \in \Omega$ (zwrotność)
- jeśli $x \equiv y$, to $y \equiv x$ (symetria)
- jeśli $x \equiv y$ i $y \equiv z$ to $x \equiv z$ (przechodność)

Klasa abstrakcji

$$[x] = \{y \in \Omega : x \equiv y\}$$

Przykład relacji równoważności

Relacja na zbiorze zwierząt

$$\Omega = \text{zbiór wszystkich ludzi}$$

$$n(\omega) = \text{płeć osoby } \omega$$

$$(\omega \equiv \eta) \Leftrightarrow (n(\omega) = n(\eta))$$

To jest relacja równoważności na zbiorze Ω.

- $[JCI] = \text{ wszyscy mężczyźni}$
- $[JSO] = \text{ wszystkie kobiety}$

Własności

- $[JCI] \cup [JSO] = \Omega$
- $[JCI] \cap [JSO] = \emptyset$

Programowanie - I

Przykład 1

```
X = ["Ala", "Jola", "Krzyś", "Emilka"];
for (i=0; i<4; i++)
  Y[i] = "Witaj " + X[i];
```

Przykład 2

```
X = [12.20, 45.50, 9.50, 21.40]
for (i=0; i<4; i++)
  Y[i] = X[i]*100;

for (i=0; i<4; i++)
  Z[i] = toString(X[i]) ++ " zł";
```

Programowanie - II

Funktor - 1

$$F(a) = \text{wszystkie ciągi skończone elementów zbioru } a$$

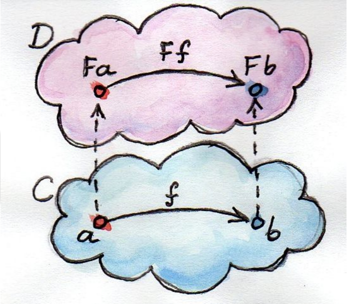
Funktor - 2

Dla $f : a \to b$ określamy

$$F(f)([a_1, a_2, \ldots, a_n]) = [f(a_1), f(a_2), \ldots, f(a_n)]$$

W języku Haskell

y = fmap f x



Programowanie - III

Przykład 1

```
x = ["Ala", "Jola", "Krzyś", "Emilka"]
y = fmap (\s -> "Witaj " ++ s) x
// y = ["Witaj Ala", "Witaj Jola", "Witaj Krzyś", ...];
```

Przykład 2

```
x = [12.20, 45.50, 9.50, 21.40]
y = fmap (\a -> a*100) x
z = fmap (\a -> toString(a) ++ " zł") x

// y = [1220, 4550, 950, 2140]
// z = ["12.20 zł", "45.50 zł", "9.50 zł", "21.40 zł"]
```

Programowanie - IV

Suma - 1

$$X = [12, 3, 5, 12, 34]$$

Suma - 2

$$S = 12 + (3 + (5 + (12 + 34))) =$$

$$12 + (3 + (5 + (12 + (34 + 0))))$$

suma

```
s=0
for (i=0; i<5;i++)
  s = s + X[i];
```

Programowanie - V

Suma i iloczyn

$$suma(X) = 12 + (3 + (5 + (12 + (34 + 0))))$$

$$iloczyn(X) = 12 * (3 * (5 * (12 * (34 * 1))))$$

Uogólnienie

$$\mathrm{foldr} \oplus \mathrm{e}[x_1, \ldots, x_n] = x_1 \oplus (x_2 \oplus x_3 \oplus (\ldots (x_n \oplus e)) \ldots)$$

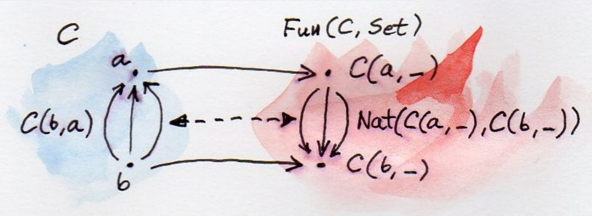
Zastosowanie

```
suma:     s = foldr (\x y -> x+y) 0 x [ = foldr (+) 0 x]
iloczyn: p = foldr (\x y -> x*y) 1 x [ = foldr (*) 1 x]
dlugość: d = foldr (\x y -> y+1) 0 x
max:      m = foldr (\x y -> max x y) 0 x
n !         = foldr (*) 1 [1..n]
```

Programowanie funkcyjne

Podstawa teoretyczna

Teoria Kategorii



Głowny język

Haskell

Zastosowanie

Wysokiej jakości kody.

To już koniec

A może nie jeszcze nie