Muzyka algorytmiczna, wykład 3

Maciej Grześkowiak

17 marca 2021

Deklaracje typów

Deklaracje typów

Deklaracja na podstawie istniejącego typu

```
type Aa = Integer
type Tabliczka = [Int]
type Dur = Rational
type AbsPitch = Int

:i Tabliczka
:i Dur
```

Deklaracje typów

Deklaracja na podstawie *całkowicie nowego* typu

```
data Bool = False | True
data Octave = Int
data PitchClass = C | Cs |...

type Pitch = (PitchClass, Octave) -- C4 = (C,4).
type Volume = Int -- 0-127
```

Deklaracje typów

```
1 data Shape = Circle Float | Rect Float Float
2 area :: Shape -> Float
3 area (Circle r) = pi * r^2
4 area (Rect x y) = x * y
```

Deklaracje typów

Deklaracja na podstawie *całkowicie nowego* typu

```
data
PitchClass = Cff | Cf | C | Dff | Cs | Df | Css | D |
             Eff | Ds | Ef | Fff | Dss | E | Ff | Es |
             F | Gff | Ess | Fs | Gf | Fss | G | Aff |
             Gs | Af | Gss | A | Bff | As | Bf | Ass |
             B | Bs | Bss
```

Typy Pitch i AbsPitch

```haskell
type Pitch = (PitchClass, Octave)
type AbsPitch = Int
```

Konwersja `Pitch <-> AbsPitch`

```haskell
pitch 60 -- (C,4)
absPitch (C,4) -- 60
```

Typy Pitch i AbsPitch

Porównywanie dźwięków (konwersja i porównanie Int)

```
1 (C,4) < (D,3)
2 (Dff, 4) == (C, 4)
3 absPitch (C,4) == absPitch(Dff,4)
```

Budowa muzyki

```
type Dur = Rational
data Primitive a = Note Dur a | Rest Dur
```

```
data Music a = Prim (Primitive a)
             | (Music a) :+: (Music a)
             | (Music a) :=: (Music a)
             | Modify Control (Music a)
```

Budowa muzyki, funkcje

```
wn, hn, qn , dqn : : Dur
```

```
c , cs , df , d :: Octave -> Dur -> Music Pitch
c 4 (1/2) ---> Prim (Note (1 % 2) (C,4))
es 3 dqn ---> Prim (Note (3 % 8) (Es,3))
```

Budowa muzyki, funkcje

```
wn, hn, qn , dqn : : Dur
```

```
note :: Dur -> a :: Music a
note qn (C,4) -- Prim (Note (1 % 4) (C,4))
```

```
rest :: Dur -> Music a
rest qn -- Prim (Rest (1 % 4))
```

Budowa muzyki, funkcje

```
line :: [Music a] -> Music a
melody :: Music Pitch
melody = line [c 5 qn , c 5 qn , g 5 qn]

Prim (Note (1 % 4) (C,5))
:+: (Prim (Note (1 % 4) (C,5))
:+: (Prim (Note (1 % 4) (G,5))
:+: Prim (Rest (0 % 1))))
```

Budowa muzyki, funkcje

```
chord :: [Music a] -> Music a
chords :: Music Pitch
chord [c 5 qn , c 5 qn , g 5 qn]

Prim (Note (1 % 4) (C,5))
:=: (Prim (Note (1 % 4) (C,5))
:=: (Prim (Note (1 % 4) (G,5))
:=: Prim (Rest (0 % 1))))
```

Budowa muzyki, funkcje

```
ps :: AbsPitch
ps = [ 60, 62, 64, 65, 67, 69, 71, 72]

majScale :: Music AbsPitch
majScale = line (map (note en ) ps

vols :: [Volume]
vols = [40,50,60,70,80,90,100,110]

majScale2 :: Music (AbsPitch , Volume)
majScale2 = line (map (note en ) (zip ps vols))
```