Dlouhá čísla

neboli

vlastní representace čísel

Dlouhá čísla

když nám nestačí standardní číselné typy...

Krok 1: Návrh representace

- a) nezáporná x i záporná
  - a2) se znaménkem x doplněk
- b) celá x desetinná
  - b2) pevná x pohyblivá řádová čárka
- c) pole x spojový seznam x string x ...
- d) odpředu x odzadu
- e) obsah jednoho prvku
- f) délka
  - f1) kolik míst „navíc“
    - spočítat (a ještě zkusit)

Dlouhá čísla [2]

Krok 2: Naprogramovat provádění operací

a) jaké operace potřebujeme

a2) jaké operace DOOPRAVDY potřebujeme

b) jak je naprogramovat

Příklad: e na 1000 desetinných míst

- odhad počtu kroků
- potřebné operace

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

$$e = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \ldots = \frac{1}{1} + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5} + \ldots$$

Příklad: e na 1000 desetinných míst

- odhad počtu kroků
- potřebné operace
  - dosazení
  - sčítání
  - dělení integerem

Příklad: e na 1000 desetinných míst

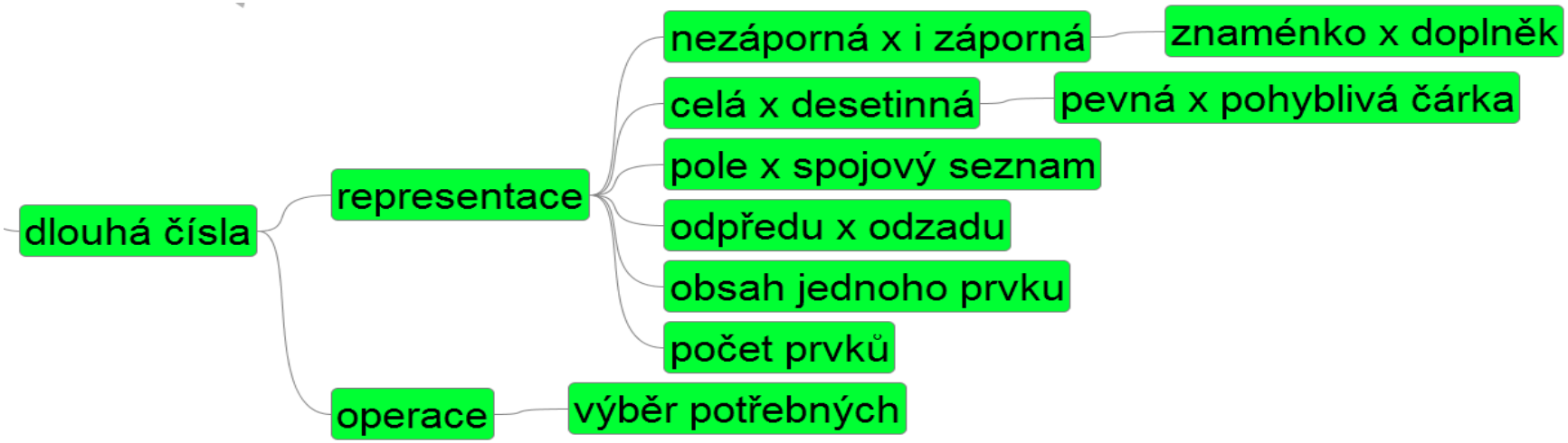
- odhad počtu kroků
- potřebné operace
  - dosazení
  - sčítání
  - dělení integerem
  - přeskočení nul
  - test konce

Jak se provádějí další operace

odečítání
- doplněk

násobení

dělení


dlouhá čísla
representace
nezáporná x i záporná
znaménko x doplněk
celá x desetinná
pevná x pohyblivá čárka
pole x spojový seznam
odpředu x odzadu
obsah jednoho prvku
počet prvků
operace
výběr potřebných

MODULY 2

Matplotlib

https://matplotlib.org

Knihovna pro 2D kreslení

Různé typy grafů

Různé výstupy

Open source

Od roku 2002

Nezisková organizace
NumFOCUS

Závislosti (potřebuje):

- Python (>= 3.6)
- NumPy (>= 1.15)
- setuptools
- cycler (>= 0.10.0)
- dateutil (>= 2.1)
- kiwisolver (>= 1.0.0)
- Pillow (>= 6.2)
- pyparsing (>=2.0.3)

https://matplotlib.org/users/installing.html

matplotlib.pyplot

sbírka funkcí pro vytváření a upravování grafu

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.show()
```

= automaticky doplněné hodnoty x

```
plt.plot([1,2,3,2,1], [2, 1, 2, 3, 2])
plt.show()
```

= zadané hodnoty x i y

```
x = list( range(360) )
y = [math.sin(x/360*2*math.pi) for x in
range(360)]
plt.plot( x,y )
plt.show()
```

matplotlib.pyplot [2]

další způsoby vykreslení:

```
plt.plot([1, 2, 3, 4], 'ro' )
plt.show()
```

obrázek (Figure) může obsahovat více grafů (plot):

```
plt.plot([1, 2, 3, 4], 'bo' )
plt.plot([2, 3, 4, 5], 'go' )
plt.plot([2, 3, 2, 5] )
```

rozsah os:

```
plt.plot([1, 2, 3, 4], 'ro' )
plt.axis([0, 10, 0, 10])
```

matplotlib.pyplot [3]

popisky os:

```
import matplotlib.pyplot as plt
plt.plot([1, 2, 3, 4])
plt.xlabel('osa x')
plt.ylabel('hodnoty')
plt.show()
```

jiné typy grafů:

```
plt.plot([1, 2, 3, 4])
plt.bar(['a','b','c','d'],[1, 2, 3, 4])
plt.show()
```

matplotlib.pyplot [4]

pod-grafy (subplot):

```
import matplotlib.pyplot as plt
plt.subplot(211)    # řádky, sloupce, číslo
plt.plot([1, 2, 3])
plt.subplot(212)
plt.plot([4, 5, 6])
plt.show()

plt.subplot(211)    # vybrat první
plt.plot([4, 5, 6], 'go')
plt.subplot(212)    # vybrat druhý
plt.plot([3, 2, 1], 'o')

plt.show()
```

matplotlib.pyplot [5]

(ne)vykreslovat postupně:

```
plt.show(block=False)
```

```
import math
import matplotlib.pyplot as plt

for i in range(100):
    plt.plot([i], [math.sin(i/10)],'ro' )
    plt.pause(0.1)
    plt.show(block=False)
plt.show()
```

matplotlib.pyplot [6]

kreslit se nemusí jenom na obrazovku:

```
plt.savefig('obrazek.png')
```

...formát pdf, svg, ps, eps...

data k vykreslování mohou být ve formátu balíčku NumPy

NumPy

balíček pro vědecké výpočty

- vícerozměrná pole
- nástroje pro zapojení kódu v C/C++ a Fortranu
- funkce pro lineární algebru, Fourierrovu transformaci a náhodná čísla

Open source

Od roku 2006

Nezisková organizace

NumFOCUS

NumPy [2]

vytvoření a naplnění pole:

```
import numpy as np
a = np.array([1,2,3,4,5,6,7])
```

prvky pole jsou stejného typu! :

```
print( a.dtype )
a[2] = 12.75  # převede na int32
b = np.array( [ [1,2], [3,4] ], dtype=complex )

   # vychozi typ = float64
```

změna typu pole:

```
af = a.astype('float64')
```

NumPy [3]

vytvoření pole s hodnotami:

```
nuly = np.zeros( (3,4) )
jednicky = np.ones( (2,3,4) )
neinicializovane = np.empty( (10,10) )

c = np.arange(20)
d = np.arange( 0, 2, 0.3 )

x = np.linspace( 0, 2*numpy.pi, 100 )
y = np.sin(x) # pro každý prvek z x vytvoří
prvek y
r = np.random.random(100) # náhodná z [0;1)
```

NumPy [4]

změna tvaru pole:

```
aa = a.reshape(4,5)
```

tisk:

- zarovnání
- i 3D pole
- ..., pokud je příliš veliké:
  ```
  print(np.arange(10000).reshape(100,100))
  ```
-

NumPy [5]

operace:

```
a2 = a+a
a3 = 10*a
aaa = a**2
mala = a<50
sb = np.sqrt(b)
```

maticové operace:

```
A = np.arange(16).reshape(4,4)
AA = A@A

N = 1000*1000
x = np.random.random(N)
y = np.random.random(N)
sum(x*x+y*y<1)/N*4
```

NumPy [6]

dosazení:

- `b = a`

  dosazení nic nekopíruje! (jen ukazatel)
- `c = a.view()` `# jiný pohled na stejná data`
- `d = a.copy()` `# tohle vytvoří kopii`

indexy:

```
a = 10*np.arange(10)
i = np.array( [2,1,3,2,7] )
a[i]
array([20, 10, 30, 20, 70])
```

NumPy [7]

Matplotlib & NumPy

Ještě trochu syntaxe

Funkce v proměnných

Funkce jako parametry funkcí

Closures