

NPRG030 Programování I

RNDr. Tomáš Holan, Ph.D.

RNDr. Martin Pergel, Ph.D.

4.patro, dveře 404

<http://ksvi.mff.cuni.cz/~holan/>

<http://kam.mff.cuni.cz/~perm>

Tomas.Holan@mff.cuni.cz

perm@kam.mff.cuni.cz

NPRG030 Programování I

- V ZS není zkouška, jen zápočet
- zkouška v LS - NPRG031 Programování II
- podmínky zápočtu určuje cvičící, ale obecně
 - aktivní účast
 - domácí úkoly
 - zápočtový test
 - zápočtový program
- zvláštní cvičení - Martin Mareš

Programování je...

- způsob, jak ovládat počítač
- umění řešit úlohy
(a počítač nám v tom může pomoci)
- umění/schopnost/dovednost psát programy

? co je to program ?

předpis, podle kterého počítač může provádět výpočet nějakého algoritmu

? co je to algoritmus ?

Příklad:

Algoritmus na sečtení dvou čísel zapsaných v desítkové soustavě.

1. Desítkové zápisy čísel umístíme pod sebe tak, aby jejich pravé okraje byly zarovnaný
2. Delší z čísel doplníme zleva jednou nulou
3. Kratší z čísel doplníme zleva tolika nulami, aby byla obě čísla stejně dlouhá
4. Postupujeme zprava a ke každé dvojici číslic určíme číslici výsledku.
Přitom tato číslice nezáleží jen na této dvojici, ale i na stavu výpočtu

- Stavů jsou dva: „s přenosem“ a „bez přenosu“
- na začátku je stav „bez přenosu“
- výsledné číslice a stav lze určit například z tabulek:

Pro stav „bez přenosu“:

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Pro stav „s přenosem“:

	0	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9	0
1	2	3	4	5	6	7	8	9	0	1
2	3	4	5	6	7	8	9	0	1	2
3	4	5	6	7	8	9	0	1	2	3
4	5	6	7	8	9	0	1	2	3	4
5	6	7	8	9	0	1	2	3	4	5
6	7	8	9	0	1	2	3	4	5	6
7	8	9	0	1	2	3	4	5	6	7
8	9	0	1	2	3	4	5	6	7	8
9	0	1	2	3	4	5	6	7	8	9

výsledek „bez přenosu“
výsledek „s přenosem“

NPR

30.09.2020 14:57:28

!!! !!! !!! !!! !!! !!! !!! !!! !!! !!! !!! !!!

ALGORITMUS NENÍ

„vysvětlit něco, co známe,
někomu, kdo to taky zná“

!!! !!! !!! !!! !!! !!! !!! !!! !!! !!! !!! !!!

Algoritmus

Posloupnost konečného počtu elementárních kroků
vedoucí k vyřešení daného typu úloh
/Encyklopedický slovník/

Charakteristické vlastnosti:

- **hromadnost** (vstupní data, výstupní data)...daného typu úloh...
- **výsledek lze získat zcela mechanicky**...elementárních kroků...
- **konečnost**...konečného počtu...

Více o algoritmech: **NPRG062 Algoritmizace**

Programovací jazyk

- Algoritmy potřebujeme zapisovat
- Přirozený jazyk se na to nehodí
- (NENÍ to jazyk „aby mu rozuměly počítače“!
Programy nepíšeme pro počítač, ale pro lidi.
)
=> programovací jazyky (mnoho a stále nové)
=> „zdrojový kód“

NPRG030: jazyk Python

NPRG031: jazyk C#

Překladač, interpret(er)... aneb nástroje

Překladač

Přečte program zapsaný v programovacím jazyku a vytvoří spustitelný program (.EXE) (=překlad).
U toho zkontroluje (syntaktickou) správnost/chybnost.

Pascal, C, C#, Java...

Interpret

Čte program zapsaný v programovacím jazyku a hned ho provádí.

Basic, JavaScript, PHP, Python...

...a různé mezi-články

Jazyk Python

- 1991, Guido van Rossum
- open source
- funguje na různých OS
- vzájemně nekompatibilní verze 2 a verze 3
- prostředí IDLE = součást instalace

Literatura

Think Python: How to Think Like a Computer Scientist

by Allen B. Downey

<http://greenteapress.com/thinkpython2/thinkpython2.pdf>

česky:

<http://howto.py.cz/index.htm>

How to Think Like a Computer Scientist

Learning with Python

by Peter Wentworth, Jeffrey Elkner, Allen B. Downey,
and Chris Meyers

<http://openbookproject.net/thinkcs/python/english3e/>

...existuje více verzí téže knihy !

Dive Into Python 3

by Mark Pilgrim

<https://diveinto.org/python3/table-of-contents.html>

česky:

Ponořme se do Pythonu 3

by Mark Pilgrim

<http://diveintopython3.py.cz/index.html>

https://knihy.nic.cz/files/edice/python_3.pdf

Co použít / nainstalovat

Python www.python.org

Verze

různé verze jazyka

drobné rozdíly

vzájemně

my budeme používat Python 3 ! (tj. verze 3.xx)

Pojďme programovat !

Základní prvky programovacího jazyka

výstup

vypsát, co jsme vypočítali

dosazení

zapamatovat si nějakou hodnotu
proměnná

vstup

přečíst zadání od uživatele

Řízení běhu programu

= rozhodování, který příkaz se teď bude provádět.

Možnosti:

- **příkaz skoku**
mění pořadí provádění příkazů
- **strukturované programování**
vytváří/skládá složitější příkazy z jednodušších

Strukturované programování

Podmíněný příkaz

if podmínka:

příkaz

příkaz

elif podmínka:

příkaz

příkaz

else:

příkaz

příkaz

Strukturované programování

Příkaz cyklu

while podmínka:

příkaz

příkaz

Které všechny příkazy se provádí, když (ne)platí podmínka a co vše se bude opakovat... určuje odsazení.

Odsazení: buďto znak tabulátor NEBO mezery.

Problém: Vypadá stejně! Zdroj chyb!

(jiné jazyky používají **begin...end**, **{...}** apod.)

Jednoduché příklady...

Co by taky šlo v Pythonu...

```
import numpy as np
import matplotlib.pyplot as plt

N = 5
menMeans = (20, 35, 30, 35, 27)
womenMeans = (25, 32, 34, 20, 25)
menStd = (2, 3, 4, 1, 2)
womenStd = (3, 5, 2, 3, 3)
ind = np.arange(N)      # the x locations for the groups
width = 0.35           # the width of the bars: can also be len(x) sequence

p1 = plt.bar(ind, menMeans, width, yerr=menStd)
p2 = plt.bar(ind, womenMeans, width,
              bottom=menMeans, yerr=womenStd)

plt.ylabel('Scores')
plt.title('Scores by group and gender')
plt.xticks(ind, ('G1', 'G2', 'G3', 'G4', 'G5'))
plt.yticks(np.arange(0, 81, 10))
plt.legend((p1[0], p2[0]), ('Men', 'Women'))

plt.show()
```

Zdroj: https://matplotlib.org/3.1.1/gallery/lines_bars_and_markers/bar_stacked.html

Jenomže...

- Nevyužívat to, co už někdo vytvořil ...je hloupost.
- Využívat cokoliv, co už někdo vytvořil ...je hloupost.

VYNALÉZAČI KOLA

(nikomu nevěřím, všechno si píšu sám...)

versus

LEPIČI A KNIHOVNÁŘI

(podívej, co jsem našel na Internetu...)

Potřebujeme rozumný kompromis...

LeftPad...

HODNOTY a PROMĚNNÉ

Hodnoty

- hodnoty vstupují, vypočítávají se, ukládají do proměnných a tisknou
- mohou být různých typů... (dále)

```
>>> type(7)
```

```
<class 'int'>
```

```
>>> type(7/2)
```

```
<class 'float'>
```

```
>>> type("ahoj")  
<class 'str'>
```

Čísla v algoritmech a programech

10^{26} Poloměr vesmíru

3063 studujících studentů MFF UK

3.142857... Ludolfovo číslo

10^{16} stáří vesmíru v sekundách !!! budeme potřebovat !!!

3 délka bakalářského studia na MFF UK v letech

36524 délka života stoletého člověka (ve dnech) (!)

...

Čísla v algoritmech a programech

=> pracujeme se dvěma*) typy čísel:

CELÁ

malý omezený rozsah (v Pythonu 3 NEOMEZENÝ!)
přesné hodnoty

DESETINNÁ

velký rozsah
přibližné hodnoty
neumíme reprezentovat všechny hodnoty

) ještě komplexní $(1+3j)(1-3j) == (10+0j)$

Typ int

neomezený rozsah!

OPERÁTORY

+ sčítání

- odčítání

* násobení

// celočíselné dělení (zaokrouhluje dolů, i pro čísla < 0)

% zbytek po dělení (periodický: $-1 \% 7 = 6$)

/ dělení s výsledkem typu float

** umocňování

==, !=, <, >, <=, >= porovnávání

Priorita operátorů

1. umocňování
2. násobení, dělení, zbytek
3. sčítání, odečítání
4. relační operátory

V případě stejné priority - odleva.

Typ float

HODNOTY

Racionální čísla, jen konečná množina,
výsledek je jen APROXIMACE správného výsledku.

VÝSLEDNÉ CHYBY záleží

na representaci čísel

na řešené úloze

na zvoleném algoritmu.

>>> $49*(1/49)$

0.9999999999999999

Odhadem velikosti chyb se zabývá **NUMERICKÁ
MATEMATIKA**.

POZOR! POZOR! POZOR! POZOR!

**V paměti je jiná reprezentace
než desítkový zápis**

=>

**při vstupu a výstupu dochází
k zaokrouhlování!**

POZOR! POZOR! POZOR! POZOR!

Poučení

Testovat reálná čísla
(vypočtená nebo načtená)
na rovnost nemusí mít dobrý smysl !

Místo toho:

$\text{abs}(\text{koruny1} - \text{koruny2}) < \text{Epsilon}$

Typ float - zápis konstant

HODNOTY

desetinná tečka a nepovinně číslice za ní
nepovinně exponent ve tvaru E<int>

SEMILOGARITMICKÝ TVAR

0.3768

-326.21

1234.

0.03E-4

-10.583E60

Funkce (pro int i float)

`int(x)`

zahodí desetinnou část

(takže pro záporná čísla zvětšuje!)

`abs(x)`

absolutní hodnota

`min(x1, x2...)`

minimum

`max(x1, x2...)`

maximum

...a další.

Funkce (pro int i float)

Ještě další funkce jsou v modulu `math`
(o modulech později), například

```
>>> import math
```

```
>>> math.factorial(69)
```

```
171122452428141311372468338881272839092270544  
893520369393648040923257279754140647424000000  
000000000
```

Typ float - operace

...jako u typu int

celočíselné dělení (zaokrouhluje dolů, i pro čísla < 0),
ale když je některý operátor typu float,
je výsledek také typu float

Proměnné

proměnná nemá svůj typ,

lze do ní uložit hodnotu jakéhokoliv typu !

```
>>> a = 2*7
```

```
>>> type(a)
```

```
<class 'int'>
```

```
>>> a = 7/2
```

```
>>> type(a)
```

```
<class 'float'>
```

Proměnné

(Na rozdíl od jiných jazyků:)

proměnná **NENÍ** místo v paměti

(jako skříňka nebo háček v šatně)

ale pouze jméno („odkaz“)

(jako lísteček, který šatnářka
přicvakne na batoh).

Proměnné (důsledek)

Dvě proměnné mohou odkazovat
na stejná data... (později).

a pokud je skrze jednu proměnnou změníme, mohou se
změnit i ve druhé proměnné

(mutable x immutable typy)

identity-operator „is“

Příklady (nebo LeftPad...)