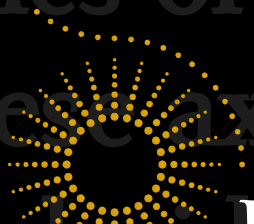
Počítače a programy
(1. přednáška)



FF UK
Department of Logic

Jonathan L. Verner

Počítač - univerzální stroj

FF UK
Department of Logic

computer, n.: a programmable usually electronic device that can store, retrieve, and process data (Merriam-Webster)

- zpracovává data na základě předpisu, který není předem dán! (Babbage)
- není (principiální) rozdíl, mezi daty a předpisy (von Neumann)!

Počítače a existencializmus

FF UK
Department of Logic

Nůž na papír je tedy současně předmětem, jenž je vyráběn určitým způsobem a který je na druhé straně jednoznačně užitečný a nelze předpokládat, že by někdo vyráběl nůž na papír, aniž by si byl vědom toho, k čemu bude tento předmět sloužit. Řekneme tedy, že podstata nože - to jest souhrn návodů a kvalit, jež ho umožňují vyrobit a určit k čemu je - . . . předchází existenci.

J. P. Sartre: Existencialismus je humanismus

Počítače a existencializmus

FF UK
Department of Logic

- Věci: podstata (esence) předchází existenci.
- Lidé: existence předchází esenci.
- Počítače: existence předchází esenci ???

Programování

FF UK
Department of Logic

programming, v.: to predetermine the thinking, behavior, or operations of a computer (Merriam-Webster)

Je třeba:

- Naučit se instrukce, kterým počítač rozumí (jazyk počítače).
- Umět přesně formulovat a analyzovat nejasně zadané problémy.
- Umět vidět problém z nadhledu a zároveň nezapomenout na detaily.
- Umět komunikovat s lidmi!
- Částečně je to umění.
- Každý (skoro) se to může alespoň trochu naučit.

Proč je dobré umět programovat

FF UK
Department of Logic

- Počítače hrají čím dál tím větší roli, je dobré vědět, co od nich lze čekat a co nikoliv.
- Programování rozvíjí analytické myšlení, problem-solving skills.
- Může to být zábavné, jako když jste si (možná) hráli se stavebnicí.
- Část logiky se dá velmi pěkně formulovat jazykem počítačových programů (Gödelovy věty, . . .)

Informatika jako věda - Computer science

FF UK
Department of Logic

- Počítač umí provést jakýkoliv proces, který jsme schopni popsat.

Jaké procesy jsme schopni (přesně) popsat?

- Návrh algoritmů/programů.
- Analýza algoritmů/programů.
- Experimentování.

Random Access Machine

FF UK
Department of Logic

- registry (AX–EX) pro krátkodobé ukládání hodnot
- speciální registr, tzv. instruction counter, ve kterém je uložena adresa příští instrukce
- instrukce pracující s registry (`CLR`, `INC`, `MOV`, `ADD`)
- instrukce pro načtení/uložení dat z/do paměti do/z registrů (`FETCH`, `STO`)
- instrukce podmíněného skoku (`JNE`) a instrukce zastavení (`HLT`)

Fibonnaciho čísla v Assembleru

FF UK
Department of Logic

```
 0: FETCH ax, 14
 1: FETCH bx, 15
 2: CLR   cx
 3: INC   cx
 4: MOV   cx, dx
 5: MOV   cx, ex
 6: ADD   ex, dx
 7: STO   ex, bx
 8: INC   bx
 9: MOV   dx, cx
10: MOV   ex, dx
11: DEC   ax
12: JNE   ax, 5
13: HLT
14: 003
15: 016
16: 000
17: 000
18: 000
```

Programovací jazyky - problémy přirozeného jazyka

FF UK
Department of Logic

Ideální by bylo, moci popsat algoritmus přirozeným jazykem.

- Počítače (zatím) nerozumí přirozenému jazyku.
- Přirozený jazyk není jednoznačný.

Viděl jsem s Petrem tři lidi.

- (A to ani nemluvím o ironii.)
- Přirozený jazyk má i jiné problémy:

Najdi nejmenší přirozené číslo, které nelze popsat méně jak dvanácti slovy.

Programovací jazyky - umělé jazyky

FF UK
Department of Logic

Nízkoúrovňové jazyky

- Strojový kód, Assembler, ...

Jazyky vyšší úrovně (kompilované)

- C, Fortran ...
- C++, Pascal, Ada, ...
- Java

Jazyky vyšší úrovně (interpretované)

- Basic, Perl, Ruby, Python

První krůčky

FF UK
Department of Logic

```
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "copyright", "credits" or "license()" for more information.
>>>
```

```
>>> print("Hello, World")
Hello, World
>>> print(2 + 3)
5
>>> print("2 + 3 =", 2 + 3)
2 + 3 = 5
>>>
```

Nové příkazy - Funkce

FF UK
Department of Logic

```
>>> def hello():
      print("Hello")
      print("Computers are fun")

>>>
```

Odsazení říká, že řádky patří k funkci hello!

```
>>> hello()
Hello
Computers are fun
>>>
```

K čemu jsou závorky?

Parametry

Příkazy/Funkce mohou mít měnící se části - parametry.

```
>>> def greet(person):
      print("Hello", person)
      print("How are you?")

>>>

>>> greet("John")
Hello John
How are you?
>>> greet("Emily")
Hello Emily
How are you?
>>>
```

Programy jako moduly

FF UK
Department of Logic

- Když zavřeme Pythonní shell, všechny příkazy se nám ztratí!
- Proto se definice ukládají do souborů, tzv. modulů (nebo skriptů).
- Tyto soubory jsou obyčejné soubory. Lze je editovat v NotePadu, dokonce, při troše (velké) námahy i ve Wordu.
- Pro editaci se ale mnohem lépe hodí speciální editory, např. IDLE.

Náš první program v Pythonu - chaos.py

FF UK
Department of Logic

```
# Soubor: chaos.py
# Jednoduchý program, který ukazuje chaotické
# chování.

def main():
    print("Program ukazuje chaotickou funkci:")
    x = float(input("Zadej číslo mezi 0 a 1: "))
    for i in range(10):
        x = 3.9 * x * (1 - x)
        print(i+1, ". člen posloupnosti je", x)

main()
```

Soubor uložíme jako chaos.py.

Náš první program v Pythonu - spouštění

FF UK
Department of Logic

- Program lze spustit různými způsoby, např. z příkazové řádky

```
Z:\home>C:\Python34\python.exe chaos.py
Tento program ukazuje chaotickou funkci:
Zadej číslo mezi 0 a 1: 0.25
1 . člen posloupnosti je 0.73125
...
```

- nebo dvojklikem na ploše (nemusí fungovat).
- Spuštením pythonu a použitím příkazu `import`, `run` nebo `execute`

```
 >>> import chaos
Tento program ukazuje chaotickou funkci:
Zadej číslo mezi 0 a 1: 0.25
1 . člen posloupnosti je 0.73125
...
```

příkaz `import`

FF UK
Department of Logic

- příkaz `import` řekne pythonu aby program načetl do paměti
- POZOR: přípona .py se nezadává!
- soubor se musí nacházet v adresáři, ve kterém byl python spušten
- při načtení python vytvoří speciální (předkompilovaný) soubor `chaos.pyc`
- po načtení jsou všechny funkce programu k dispozici pomocí tečkového zápisu

```
>>> chaos.main()
Tento program ukazuje chaotickou funkci:
Zadej číslo mezi 0 a 1:
...
```

chaos.py - čím je tvořen 0.

FF UK
Department of Logic

První tři řádky začínají znakem `#`, jsou určeny pro čtenáře, python je ignoruje

```
# Soubor: chaos.py
# Jednoduchy program, ktery ukazuje chaoticke
# chovani.
```

Pak následuje definice funkce s názvem `main`:

```
def main():
```

Technicky vzato to nebylo nutné, avšak tradičně se instrukce, ze kterých je program složen, vkládají do funkce zvané `main`. Výhodou je, že lze program znovu spustit pomocí tečkové notace: `chaos.main()`

chaos.py - čím je tvořen 1.

FF UK
Department of Logic

Další řádka je vlastně začátkem našeho programu. Vypíše kratičkou zprávu na obrazovku:

```
print("Program ukazuje chaotickou funkci:")
x = input("Zadej číslo mezi 0 a 1: 0.25: ")
```

- `x` je příkladem proměnné (jména)
- Proměnné se používají, když si chceme označit nějakou hodnotu, abychom se na ni mohli později odkázat.
- Python vypíše `Zadej číslo mezi 0 a 1:` a čeká na uživatele
- To, co uživatel zadá je uloženo jako proměnná `x`

chaos.py - čím je tvořen 2.

FF UK
Department of Logic

```
for i in range(10):
```

- Tato řádka je příkladem cyklu.
- Cyklus je způsob, jak provést sadu příkazů pro každý člen nějaké posloupnosti.
- Tento konkrétní cyklus provádí následující odsazené řádky pro hodnoty `i` od 0 do 9:

```
    x = 3.9 * x * (1 - x)
    print(i+1, ". člen posloupnosti je", x)
```

Stejného efektu bychom docílili, kdybychom tyto řádky napsali 10x pod sebe:

```
x = 3.9 * x * (1 - x)
print(1, ". člen posloupnosti je", x)
x = 3.9 * x * (1 - x)
print(2, ". člen posloupnosti je", x)
... (8x)
```

chaos.py - čím je tvořen 3.

FF UK
Department of Logic

```
x = 3.9 * x * (1 - x)
```

- Tento řádek je příkladem přiřazení (definice).
- Na pravé straně = je matematický výraz: `3.9 * x * (1-x)`
- Pokud hodnota `x`, kterou uživatel zadal, byla např. 0.25, vidíme, že tento výraz má hodnotu $3.9 \times 0.25 \times (1 - 0.25)$, což je 0.73125.
- Jméno (proměnná) `x` nyní označuje (odkazuje na, pojmenovává) hodnotu , která je napravo od =, v našem případě hodnotu výrazu `3.9 * x * (1-x)`
- Pokud tedy `x` prve označovalo 0.25 po provedení tohoto příkazu bude nově označovat 0.73125.

Co udělá další řádek už víme:

```
print(i+1, ". člen posloupnosti je", x)
```

Vypíše hodnotu, na kterou právě odkazuje jméno `i`, text ". člen..." a hodnotu na kterou odkazuje `x`, tedy 0.73125.

chaos.py - čím je tvořen 4.

FF UK
Department of Logic

Tento proces se ale ještě 9x zopakuje. Avšak hodnota `x` se stále mění. Například při druhém průchodu, bude na pravo od = matematický výraz $3.9 \times 0.73125 \times (1 - 0.73125)$, což je 0.76644140625.

Chaos

FF UK
Department of Logic

- Program vypisuje prvních deset členů speciálního případu tzv. logistické posloupnosti.
- Tato posloupnost je definována rekurzivním vzorečkem:

$$a_{n+1} = k \cdot a_n \cdot (1 - a_n)$$

- Modeluje chování některých nestabilních el. obvodů nebo populací za omezujících podmínek.
- Posloupnost je zdánlivě nepředvídatelná.
- Je citlivá na počáteční hodnotu - efekt motýlích křídel.
- !!! Chaos $\neq$ Náhodnost !!!
- Spojitost s dynamickými systémy.

Alternativa k chaos.py

FF UK
Department of Logic

```
# Soubor: chaos.py (alternativní verze)
# Jednoduchý program, který ukazuje chaotické
# chování.

def lg(k, x):
    return k * x * (1 - x)

def main():
    print("Program ukazuje chaotickou funkci:")
    x = float(input("Zadej číslo mezi 0 a 1: "))
    for i in range(10):
        x = lg(3.9, x)
        print(i+1, ". člen posloupnosti je", x)

main()
```

Addendum - instalace Pythonu

FF UK
Department of Logic

`www.python.org`

- Sekce download
- Je třeba instalovat verzi 3.něco

`https://www.python.org/ftp/python/3.5.1/python-3.5.1-amd64.exe`