Programování 1

Martin Pergel, perm@kam.mff.cuni.cz

26. listopadu 2019

Informace o přednášce a cvičeních

- Kurz je zakončen zápočtem, zkouška bude v létě.
- Podmínky zápočtu:
  - Praktický zápočtový test,
  - zápočtový program (domácí práce),
  - aktivní účast (body v CodExu),
  - další dle požadavků cvičících.

Cíle předmětu:

- Vývojové prostředí (Visual Studio .Net),
- jazyky Python a C#,
- algoritmy (outsourceovány do Algoritmizace, leč budou využity),
- a teorie s tím související.

Jednotlivé položky budou probírány paralelně!

Python

a vývojová prostředí

- python3 (přímo interpret),
- IDLE (jednoduché prostředí),
- ale pro naše účely až příliš jednoduché,
- Visual Studio .NET využijeme i v létě.
- Prostředí vyberte dle vlastního uvážení, ostatní budou trpěna (ne systematicky podporována).

Organizační záležitosti

- Stroj ReCodEx (alias ReCodEx Code Examiner a účet na něm)
- Účty v příslušných počítačových učebnách (Malá Strana a další budovy)
- Existují dvě paralelní přednášky, které jsou ekvivalentní, nikoliv totožné.
- Pozor, programování je dovednost náročná na čas!

Literatura

- A. B. Downey: Think Python: How to Think Like a Computer Scientist (http://greenteapress.com/thinkpython2/thinkpython2.pdf, česky http://howto.py.cz/index.htm),
- P. Wentworth, J. Elkner, A. B. Downey, Ch. Meyers: How to Think Like a Computer Scientist Learning with Python (http://openbookproject.net/thinkcs/python/english3e/) existuje více verzí!
- M. Pilgrim: Dive Into Python 3 / Ponořme se do Pythonu 3 (https://diveinto.org/python3/table-of-contents.html, http://diveintopython3.py.cz/index.html)
- Dotazy? [pokud ano, ptejte se ihned]

Algoritmy:

Definition

Algoritmy jsou přesně definované a záměrně vytvořené postupy.

- Algoritmus je postup, jak řešit určitý problém.
- Realizací algoritmu dojdeme od zadaných (vstupních) dat k požadovanému výsledku.
- Sestává z "kroků" zvaných příkazy (příklad - alg. sčítání čísel).
- Správný algoritmus musí být:
  - konečný (pro každý vstup doběhne v konečném čase)
  - a parciálně správný (pokud doběhne, odpoví správně).

Způsoby zápisu algoritmu

Karel:

```
krok
krok
krok
vlevobok
krok
```

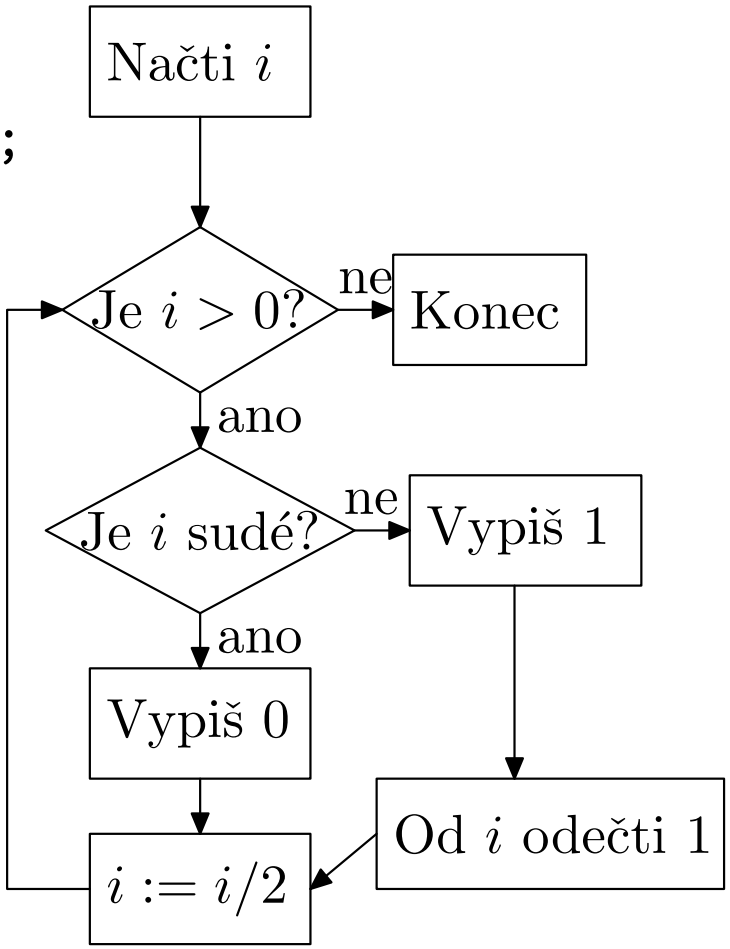
Jazyk C:

```
while(i)
{ if(i%2) printf('1');
  else printf('0');
  i/=2;
}
```

Text:

Načti hodnotu i.
Dokud je $i > 0$:
    Je-li je i liché vypiš "1"
    jinak vypiš "0"
    Vyděl i dvojkou.

Vývojové diagramy:



Dvojková soustava

- Čísla zapisujeme v desítkové soustavě ve tvaru $a_n a_{n-1} \ldots a_1 a_0$,
- význam: $\sum_{i=0}^{n} a_i \cdot 10^i$.
- Potřebujeme deset číslic $(0 - -10)$.
- Ve dvojkové soustavě je význam $\sum_{i=0}^{n} a_i \cdot 2^i$,
- číslice stačí 2.

Rozklad na prvočinitele

- Nápady
- Naivní algoritmus: Hledej postupně prvočísla až do n a zkoušej jimi dělit.
- Pokus o méně naivní algoritmus: `Hledej prvočísla do` $\sqrt{n_i}$ `(kde` n_i `je hodnota, kterou zbývá faktorizovat).` nebude fungovat. Proč?
- Méně naivní algoritmus funkční: `Hledej prvočísla do` $\sqrt{n_i}$ `(kde` n_i `je hodnota, kterou zbývá faktorizovat). Zbývá-li více než 1, vypiš zbývající číslo.` Proč toto funguje?

Zápis programů

Při programování (zápisu algoritmů):

- pracujeme s proměnnými různých typů (a s konstantami),
- modifikujeme obsahy proměnných,
- voláme podprocedury,
- porovnáváme obsahy proměnných,
- rozhodujeme se podle toho,
- cyklíme,
- čteme vstup, vypisujeme výstup.

Proměnné a jejich typy:

Proměnná má implicitně přiřazený typ.
Primitivní datové typy (v Pythonu) jsou:

- **integer:** celé číslo,
- **float:** necelé číslo,
- **string:** řetězec znaků (text),
- **boolean:** pravda nebo lež (nabývá hodnot `True` a `False`).
- Typy jsou implicitní, dávejte na ně pozor (1 vs "1").
- Další typy (seznamy, třídy,...) budou později.
- Mezi **string**em a **int**em převádějí funkce **str** a **int**.
- Příklad načtení stringu a čísla.

Výrazy aritmetické:

- $+$ Sčítání,
- $-$ odčítání,
- $*$ násobení,
- $/$ dělení,
- závorky,
- `//` celočíselné dělení,
- `%` zbytek po dělení (ve stringu substituce).
- Pozor: `'nazdar '*10`

Přiřazovací příkaz: `=`

Příklad: `x= 2*y;`

Relační operátory

- $<$ je menší než (kupř. $a < b$),
- $>$ je větší než,
- $>=$ je větší nebo rovno,
- $<=$ je menší nebo rovno,
- $!=$ nerovná se,
- $==$ rovná se (porovnání na rovnost).

Podmínky

Syntax (a sémantika):

- `if` podminka `:`
  prikazy
  takto_odindentovane
- `if` podminka `:`
  prikazy
  else:
  prikazy

Příklad:

```
if teplota>25 :
        print('Jdu do hostince!')
```

Příklad

```
if teplota>25:
    print('Jdu do hostince!')
```

```
if teplota>25 :
begin
        print('Jdu do hostince!')
else:
        print('Nejdu nikam!')
```

Prostředí pro práci s Pythonem

- Python je interpretovaný jazyk, ke spuštění kódu je třeba interpret.
- Alternativou jsou překládané jazyky, kdy je třeba překladač (a výstup pak lze samostatně spustit).
- V Pythonu lze zdrojový kód také zkompilovat, ale to dělat nebudeme.
- Předvést interpret (python3), IDLE a Studio.