Úvod do programovacích jazyků

doc. Dr. Ing. Miroslav Beneš
katedra informatiky, A-1007
☎ 59 732 4213

Obsah předmětu

- **Funkcionální programování**
  - Programovací jazyk Haskell
  - Abstraktní datové typy
  - Rekurze
- **Teorie programovacích jazyků**
  - Specifikace jazyků – syntaxe a sémantika
  - Základní jazykové konstrukce
- **Skriptovací jazyky**
  - Programovací jazyk PHP

Požadavky

- Písemná práce – 20 bodů
  - funkcionální programování
  - konec dubna
- Projekt – 2 x 15 bodů
  - Haskell
  - PHP
- Zkouška – 50 bodů
  - Test na témata z přednášek

min. 26 bodů pro zápočet

Projekt

- Řešení příkladu na zvolené téma
  - v jazyce Haskell
  - v jazyce PHP
- Termín odevzdání 26. května 2003
- Témata jsou k dispozici na stránkách
  http://www.cs.vsb.cz/benes/vyuka/UDPJ

Klasifikace programovacích jazyků

- Imperativní jazyky
  - posloupnosti příkazů + řídicí struktury
  - program má implicitní *stav*, který se modifikuje konstrukcemi programovacího jazyka
  - explicitní pojem *pořadí* příkazů
  - vyjadřuje, *jak* se má program vyhodnocovat
  - vhodné pro tradiční architektury počítačů (von Neumann)
  - většina současných jazyků – Pascal, C, Java

Klasifikace programovacích jazyků

- Deklarativní jazyky
  - program *nemá* implicitní stav, stavové informace se udržují explicitně
  - program je tvořen *výrazy* (termy), ne příkazy
  - opakované provádění se vyjadřuje *rekurzí*
  - vyjadřujeme, *co* se má spočítat
  - funkcionální a logické programovací jazyky, dotazovací jazyky, …

Deklarativní jazyky

- funkcionální (aplikativní) jazyky
  - základním modelem výpočtu je matematický pojem funkce
  - aplikované na argumenty a vypočítávající deterministicky jediný výsledek
  - vycházejí z teorie funkcí - λ-kalkulu
  - **čistě funkcionální jazyky**: FP Haskell Miranda Hope Orwell
  - **hybridní jazyky**: Lisp Scheme SML
  - **dataflow jazyky**: Id Sisal

Deklarativní jazyky

- relační (logické) jazyky
  - základním modelem výpočtu je pojem *relace* (předikát)
  - vycházejí z teorie predikátového počtu
  - výpočet probíhá nedeterministicky s *navracením* (backtracking)
  - Prolog, Parlog, Datalog, Goedel

Historie funkcionálních jazyků

- 1930 Alonzo Church
  - vytvoření (netypovaného) λ-kalkulu
  - práce v oblasti teorie funkcí a vyčíslitelnosti (Churchova teze)
- 1958 MacCarthy
  - vytvoření jazyka LISP (List Processing)
  - „Lost In Stupid Parentheses“
- 1965 P. Landin
  - jazyk ISWIM, předchůdce jazyků typu ML

Historie funkcionálních jazyků

- 1978 R. Milner
  - jazyk ML, metajazyk pro systém podporující dokazování
- 1989 A. Appel, D. MaxQueen
  - jazyk Standard ML (SML)
- 1985 D. Turner
  - jazyk Miranda (komerční)
  - jazyk s líným vyhodnocováním
- 1987--1997 Haskell
  - plně modulární funkcionální jazyk s líným vyhodnocováním
  - volně dostupné implementace

Důvody pro studium funkcionálního programování

- S funkcionálními programy se lépe pracuje
  - *referenční transparence*
  - v daném kontextu představuje proměnná vždy tutéž hodnotu

```
{ x = 3 }
x := x + 1
{ x = 4 }
```

  - "stejné lze nahradit stejným"
    - `x - x` ↮ `f(1) - f(1)`
- FP jazyky mají lepší mechanismus abstrakce
  - abstrakce chování (algoritmu) pomocí *funkcí vyššího řádu*
  - funkce jako "first-class" hodnota
    - $\sum_{i=1..n} x_i = 0 + x_1 + x_2 + \ldots$
    - $\prod_{i=1..n} x_i = 1 * x_1 * x_2 * \ldots$
    - $compute(C, (+), \mathbf{x}) = C\ (+)\ x_1\ (+)\ x_2\ (+)\ \ldots$

Důvody pro studium funkcionálního programování

- Díky možnosti abstrakce chování jsou programy stručnější
  - PROBLÉM: kryptografické programy
- FP umožňuje nové algoritmické přístupy
  - *lazy evaluation x eager evaluation*
  - práce s potenciálně nekonečnými datovými strukturami
  - oddělení dat od řízení (nemusíme se starat o průběh vyhodnocení)

Důvody pro studium funkcionálního programování

- FP umožňuje nové přístupy k vývoji programů
  - dokazování programů
  - transformace programů na základě algebraických vlastností
- FP umožňují využití (masivně) paralelního výpočtu
  - jednoduchá dekompozice programu na části, které lze vyhodnocovat paralelně
  - PROBLÉM: paralelismu je potenciálně příliš mnoho

Důvody pro studium funkcionálního programování

- FP je důležité pro některé oblasti aplikace informatiky
  - umělá inteligence
  - specifikace, modelování
  - rychlé prototypování
- FP má blízký vztah k teoretické informatice
  - sémantika programovacích jazyků
  - teorie složitosti algoritmů a vyčíslitelnosti

Důvody pro studium funkcionálního programování

- FP je zajímavou a myšlení rozvíjející činností pro studenty informatiky!
  - jiný pohled na programování