برنامه نویسی تابعی

Functional Programming

تهیه کنندگان:
حسن اسلامی (۸۶۱۰۰۵۴۶)
امیر شیخها (۸۶۱۰۵۳۷۴)

انواع مدل‌های برنامه‌نویسی

- تقسیم‌بندی‌های مختلف
  - ساخت یافته (structured) و فاقد ساخت (unstructured)
    - ساخت یافته شامل
      - پیمانه‌ای (modular)
      - شی گرا (object oriented)
      - ....
    - فاقد ساخت شامل
      - برنامه‌نویسی آرایه‌ای
      - برنامه‌نویسی تکراری
      - ....

انواع مدل‌های برنامه‌نویسی

- تقسیم‌بندی‌های مختلف
  - امری (imperative) و توصیفی (declarative)
    - امری شامل
      - برنامه‌نویسی رویه‌ای (procedural)
      - ....
    - توصیفی شامل
      - تابعی (functional)
      - منطقی (logic)
      - ....

برنامه‌نویسی توصیفی

- بیان منطق محاسبات
  - در مقابل برنامه‌نویسی امری : بیان جریان کنترلی برنامه (الگوریتم)
- برنامه باید "چه" کار کند
  - در مقابل برنامه‌نویسی امری : برنامه "چگونه" کار می‌کند
- referential transparency: اجرای برنامه عموماً بدون اثر جانبی
  - اثر جانبی (side effect) : تغییر در حالت برنامه
    - تغییر متغیر سراسری
    - ایجاد استثنا (exception)

برنامه‌نویسی تابعی (مفهوم)

- ساخته شده مبتنی بر حساب لامبدا (**λ-calculus**)
  - سیستم رسمی برای
    - توصیف توابع
    - بیان کاربردهای توابع
    - توابع بازگشتی

برنامه‌نویسی تابعی (مفهوم)

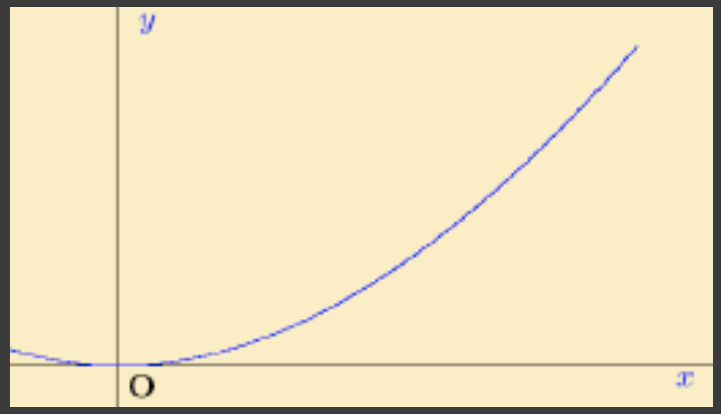
- یک تابع
  - زوج‌های مرتب ورودی و خروجی

(1,1) (2,4) (3,9) (4,16) …

  - یک رابطه توصیف کننده

f x = x^2

  - یک نمودار



برنامه‌نویسی تابعی (مفهوم)

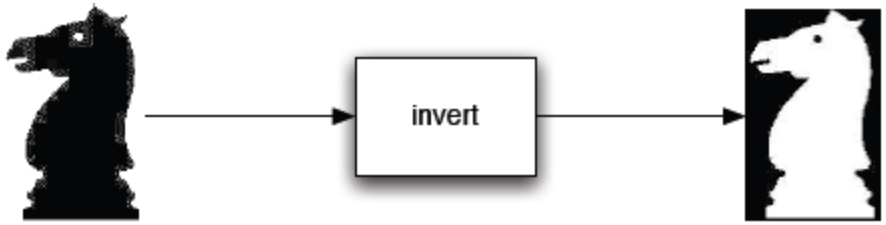
- انواع داده‌های یک تابع
  - Integer
  - Floats
  - Characters
  - Strings
  - Pictures
  - ....

برنامه‌نویسی تابعی (مفهوم)

- مثال (تعریف یک تابع ساده)

```
invert :: Picture -> Picture
knight :: Picture

invert knight
```

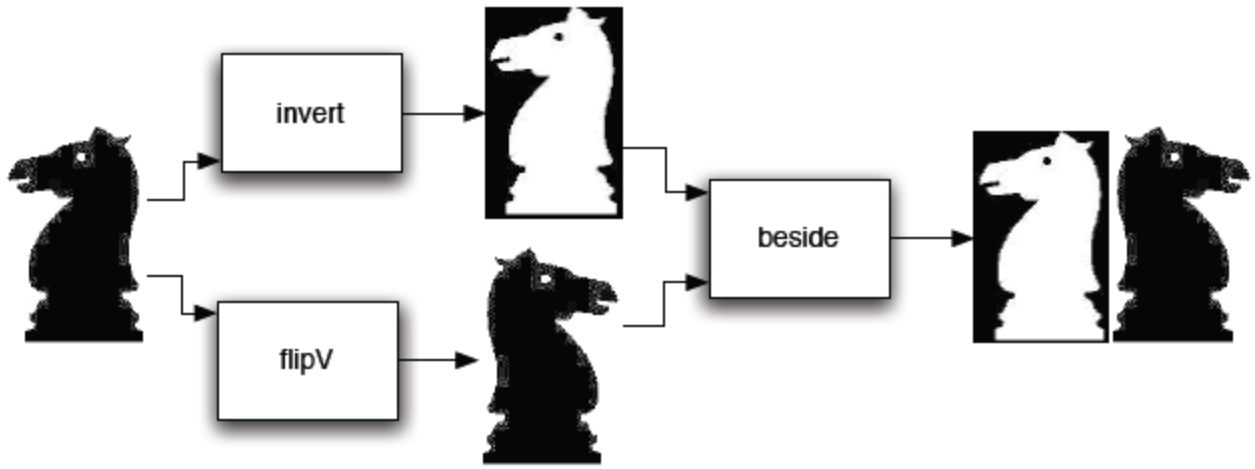


برنامه‌نویسی تابعی (مفهوم)

- مثال (ترکیب توابع)

```
beside :: Picture -> Picture -> Picture
flipV :: Picture -> Picture
invert :: Picture -> Picture
knight :: Picture

beside (invert knight) (flipV knight)
```

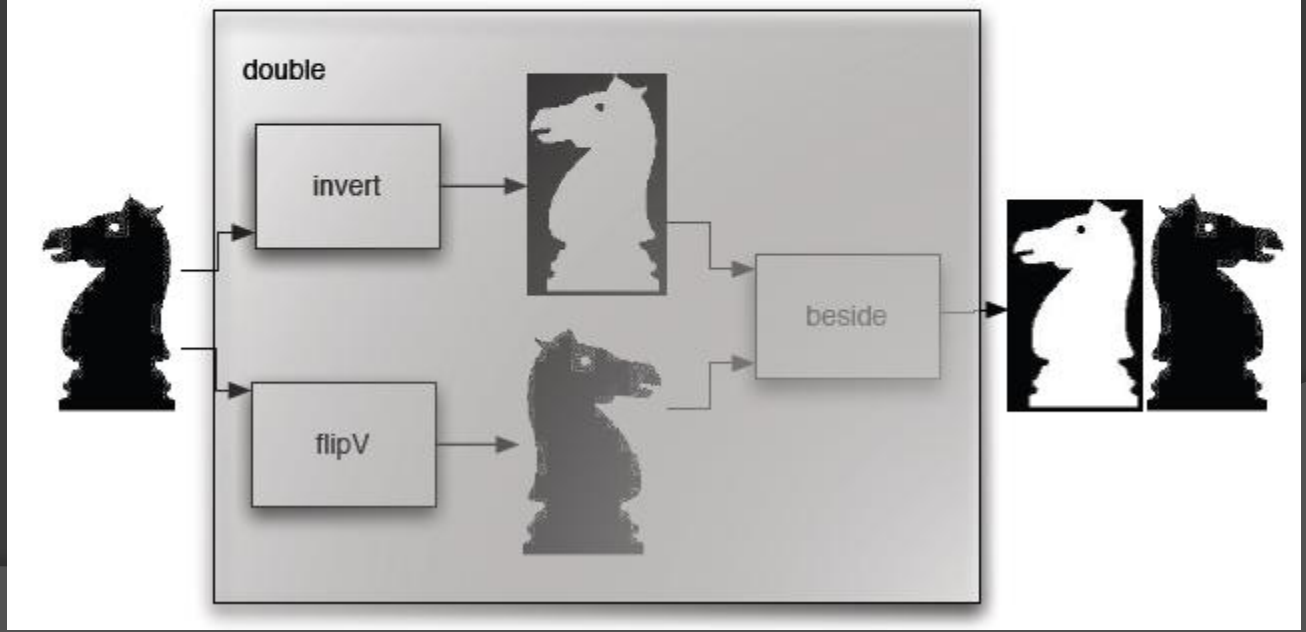


برنامه‌نویسی تابعی (مفهوم)

- مثال (تعریف تابع جدید)

```
double :: Picture -> Picture
double p  =  beside (invert p) (flipV p)

double knight
```

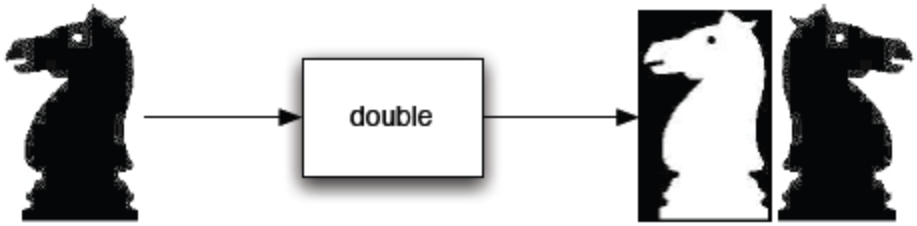


برنامه‌نویسی تابعی (مفهوم)

- مثال (تعریف تابع جدید)

```
double :: Picture -> Picture
double p  =  beside (invert p) (flipV p)

double knight
```



برنامه‌نویسی تابعی (تاریخچه)

- اولین زبان تابعی
  - LISP توسط Jonh McCarthy
- و در ادامه
  - IPL : اولین زبان تابعی برای کامپیوترهای شخصی
  - APL
  - FP : ایجاد سلسله مراتب در برنامه‌های تابعی
  - ML : مسندات بزرگ و پیچیده ریاضی
  - Haskell : استاندارد اما با شکل‌های مختلف (open standard)

تفاوت با زبان‌های امری

- توابع در زبان‌های امری اثرات جانبی دارند
  - می‌توانند باعث تغییر حالت برنامه شوند
  - مقادیر یکسان ورودی به توابع در زمان‌های مختلف نتیجه مختلف دارند
- توابع در زبان‌های تابعی referential transparency دارند
  - همواره می‌توان مقدار تابع را به جای خود تابع در یک عبارت گذاشت
  - نتیجه مهم: پیش‌بینی و درک ساده‌تر رفتار برنامه

مفاهیم اساسی

- توابع مرتبه اول و مرتبه بالاتر
  - تابع مرتبه بالاتر می‌تواند به عنوان ورودی یا خروجی یک تابع را بگیرد یا بدهد
    - مشتق و انتگرال (!)

```
double = beside (invert) (flipV)
```

  - تابع مرتبه اول می‌تواند در جای سایر عناصر مرتبه اول (عدد و ...) بیاید

```
double p = beside (invert p) (flipV p)
```

مفاهیم اساسی

- توابع خالص
  - عدم وجود اثرات جانبی حافظه یا ورودی/خروجی
- بازگشت
  - تمام حلقه ها تبدیل به توابع بازگشتی

Haskell

By: Hassan Eslami
Amir Shaikhha

History

- It is named after logician Haskell Curry
- the language is rooted in the observations of Haskell Curry and his intellectual descendants:

> "a proof is a program; the formula it proves is a type for the program"
>
> Haskell Curry

History

- Following the release of Miranda by Research Software Ltd, in 1985, interest in lazy functional languages grew
- by 1987, more than a dozen non-strict (lazy), purely functional programming languages existed
- Miranda was the most widely used, but was not in the public domain

Haskell Versions

- **Haskell 1.0**
  - was defined in 1990
  - The scientists' efforts resulted in a series of language definitions
- **Haskell 98**
  - intended to specify a stable, minimal, portable
  - an accompanying standard library for teaching, and as a base for future extensions

Haskell Versions

- **Haskell Prime**
  - In 2006,
  - revise the language definition
- **Haskell 2010**
  - allowing for bindings to other programming languages
  - fixes some syntax issues (changes in the formal grammar)

Haskell is

- **Purely** functional
  - a function is a first-class citizen
- Non-strict

- With a *clear elegant* syntax
  - So-called “Syntactic Suger”

Features

- Lazy Evaluation
- Pattern Matching
- List Comprehensions
- Type Polymorphism
- Type Class
- Built-in Memory Management
  - Garbage Collection

Strict programming language

- Only strict functions
  - functions whose parameters must be evaluated completely before they may be called
- non-strict programming language
  - non-strict functions, and hence may allow lazy evaluation

Lazy Evaluation

- delaying a computation until the result is required
  - performance increases
  - avoiding error conditions in the evaluation of compound expressions
  - constructing potentially infinite data structures
- evaluation strategy
  - call-by-name
  - call-by-need (Haskell)

Lazy Evaluation

- like Unix pipes

  grep printf Foo.c | wc

  “wc” demanding “lines” from “grep”

  But:

  grep printf Foo.c | head 5

  Can you appreciate the difference?

List Comprehension

- Syntactic construct available for creating a list based on existing lists
- Math Notation

$$S = \{ 2 \cdot x \mid x \in \mathbb{N},\ x^2 > 3 \}$$

- Haskell Code (!!)

```
s = [ 2*x | x <- [0..], x^2 > 3 ]
```

Haskell vs. C

- C is
  - Faster
  - Less memory usage
- Haskell, in contrast,
  - Less effort in programming
  - Implicit memory management

Applications

- Increasingly being used in **commercial situations**
- Darcs
  - revision control system written in Haskell
- Linspire
  - GNU/Linux chose Haskell for system tools development
- Xmonad
  - window manager for the X Window System, written entirely in Haskell

Applications

- Non-commercial usage
  - Operating system
    - Smart-card
  - Database system
    - Kleisli/CPL – The Kleisli Query System
    - Froglingo – database and programming language
  - Games
    - Cherry: a chess processor
    - The Drops Juggling Animator

Relative Languages

- Epigram
- Agda
- Curry
- Jaskell
- Parallel Haskell
  - supports clusters of machines or single multiprocessors.
  - support for Symmetric Multiprocessor parallelism.

Relative Languages

- Distributed Haskell
- Generic Haskell
- O'Haskell

Criticism

- Problems with lazy evaluation
  - improved performance in practice
  - laziness makes it more difficult for programmers to **reason about the performance of their code**
  - particularly its **space usage** is not determined
- Hard for Haskell learners:
  - "The subtle syntax and sophisticated type system of Haskell are a double edged sword — **highly appreciated by experienced** programmers but also a **source of frustration among beginners**, since the **generality of Haskell often leads to cryptic error** messages."

References

- MacLennan B. J., ***Principles of Programming Languages: Design, Evaluation, and Implementation***, 3rd edition.
- Friedman D., Wand M., ***Essentials of Programming Languages***, 3rd Edition, MIT Press, 2008.
- http://en.wikipedia.org/wiki/Functional_programming
- http://www.haskell.org/haskellwiki/Functional_programming
- http://c2.com/cgi/wiki?FunctionalProgramming
- Thompson S., ***Haskell: The Craft of Functional Programming***, 2nd Edition, Addison-Wesley, 1999.

سوالات

- سه تفاوت زبان‌های تابعی با زبان‌های امری را بیان کنید.
- در مورد کاربرد زبان‌های تابعی برای نرم‌افزارهای بزرگ (enterprise application) توضیح دهید. آیا برای اینگونه برنامه‌ها مناسبند؟ چرا؟
- زبان برنامه‌نویسی Haskel را با زبان C مقایسه کنید.

با تشکر