Structuur van Computerprogramma's 1

HET TENTAMEN

2023

Het schriftelijk tentamen

vaste dag; kijk op je lesrooster

3 uur op papier

3/4/5 vragen, zowel theorie als programmeeroefeningen

kijk naar voorbeelden van de vorige jaren!

let op: lesvolgorde is veranderd !

Wat moet ik kennen voor het tentamen?

ALLES in de lessen 1-2-3

les 4 tot aan slide 22

les 5 tot aan side 28

(1) wat moet ik kunnen doen?

cfr. oefeningen

(2) wat moet ik kunnen uitleggen?

Les 1

Proceduredefinitie, procedureoproep

Formele en actuele parameters

Call-by-value parameterpasseermechanisme

Omgevingsmodel voor procedure evaluatie

Procedurele abstractie

Les 2

De methode van Newton (benaderingsmethode)

Locale procedures en blokstructuur

Vrije en gebonden variabelen

Lexicale scoop

Les 3

Procedures <> Processen

Lineair recursieve processen

Lineair iteratieve processen

Lineaire recursie <> iteratie

Definitie van een iteratief proces

Boomrecursieve processen (exponentieel)

Logaritmische processen

Les 4

Eerste klas objecten; Procedures zijn eerste klas objecten

Wat is dan een hogere orde procedure

Patronen abstraheren in een hogere orde procedure (veralgemeende som)

Anonieme procedures construeren met lambda

De 2 gezichten van define

Locale variabelen; let, let*

~~Nulpunten berekenen met de half-intervalmethode~~

~~Fixpunten berekenen~~

~~Wortel berekenen als een fixpunt; demping~~

~~Procedures als terugkeerwaarde~~

~~Demping iha~~

Les 5

Cons cellen en lijsten

Lineair recursieve processen over vlakke lijsten

Hoger orde lineair recursieve processen over vlakke lijsten
(map-accumulate-filter)

~~Bomen als geneste lijsten~~

2 Vraag 2: Recursie, Iteratie, en Hogere Orde

(a) Als je weet dat het aantal besmettingen met het ABC virus elke dag verdubbelt, schrijf dan een procedure `(bereken-besmettingen n t)` uitrekenen hoeveel besmettingen er zijn na `t` dagen als je start met `n` besmettingen.

```
> (bereken-besmettingen 1024 3)
8192
```

(b) Genereert je oplossing een recursief of iteratief proces? Leg kort in je eigen woorden uit waarom je dit antwoord geeft, en schrijf een versie die het andere proces genereert.

(c) Andere virussen kennen typisch een heel ander verloop. Het aantal nieuwe besmettingen na 1 dag kan gegeven worden door een (ingewikkelde) functie $F(n, p)$ waarbij n het aantal besmettingen is en p de grootte van de totale populatie. Schrijf een hogere orde procedure die, gegeven een besmettingsfunctie `f` voor een willekurig virus, kan uitrekenen hoeveel besmettingen er zijn na `t` dagen als je start met `n` besmettingen op een totale populatie van `p`.

(d) Herschrijf je procedure uit **(a)** door gebruik te maken van je nieuwe hogere orde procedure uit **(c)**.

3 Recursie en iteratie op lijsten

a) Schrijf een procedure `(merge lst1 lst2)` die twee lijsten samenvoegt tot een enkele lijst door telkens 1 element van de eerste lijst te laten volgen door 1 element van de tweede lijst, enzovoort. Als 1 van de 2 lijsten korter is dan de andere vul je gewoon aan met de overblijvende elementen van de langste lijst.

```
> (merge '(1 2 3 4 5) '(6 7 8 9 10))
(1 6 2 7 3 8 4 9 5 10)
> (merge '(1 2 3) '(4 5 6 7 8 9 10))
(1 4 2 5 3 6 7 8 9 10)
> (merge '(a b c) '())
(a b c)
```

b) Levert jouw implementatie voor `merge` een recursief of een iteratief proces op? Leg in je eigen woorden uit waarom je dit antwoord geeft. Schrijf nu ook de andere versie van `merge`.

4 Vraag 4: Omgevingsmodel

Drie studenten werden gevraagd om een procedure `(oefening x y)` te schrijven die $(x - y)^3$ berekent, en kwamen elk met een verschillende oplossing op de proppen.

(1) Kwik schreef zijn procedure als volgt:

```
(define (oefening x y)
  (define (cube x)
    (* x x x))
  (let ((hulp (- x y)))
    (cube hulp)))
```

(2) Kwek, die achter Kwik zat, en zijn scherm een beetje kon zien, schreef het iets anders:

```
(define (oefening x y)
  (define (hulp)
    (- x y))
  (define (cube x)
    (* x x x))
  (cube (hulp)))
```

(3) Jantje, die achter Kwek zat, en toevallig ook verziend is, schreef het dan weer zo:

```
(define (oefening x y)
  (define (cube x)
    (* x x x))
  (let ((hulp (- x y))
        (result (cube hulp)))
    result))
```

Welke oplossingen zijn correct, en welke niet? Teken voor elke oplossing een klein omgevingsmodel voor de oproep `(oefening 3 7)` en gebruik je omgevingsmodel om te argumenteren waarom de oplossing al dan niet correct is.

1 Theorie

Onderstaande code, die we in de cursus bestudeerd hebben, berekent de wortel van een gegeven getal.

```
(define (sqrt x)
  (define (good-enough?  guess)
    (< (abs (- (square guess) x)) 0.001))
  (define (improve guess)
    (average guess (/ x guess)))
  (define (sqrt-iter guess)
    (if (good-enough? guess)
        guess
        (sqrt-iter (improve guess))))
  (sqrt-iter 1))
```

a) Leg in je eigen woorden uit waarom deze methode een *benaderingsmethode* is.
b) Leg uit waarom het in de `good-enough?` hulpfunctie niet voldoende is om `(< (- (square guess) x) 0.001)` te gebruiken.
c) Wat zijn in de definitie van de `improve` functie de gebonden en de vrije variabelen? Wat is het taalmechanisme dat maakt dat je in de body van de `improve` functie de parameter `x` van de hoofdfunctie mag gebruiken?
d) Veronderstel dat (alle andere code identiek) de `improve` functie vernederlandst wordt naar

```
(define (improve gok)
  (average gok (/ x gok)))
```

gaat het geheel nog werken?