

Lista zagadnień nr 1

Przed zajęciami

Pierwsza lista zagadnień poświęcona jest podstawom programowania w Rackecie. Przed zajęciami należy przeczytać ze zrozumieniem **Rozdział 1.1** podręcznika; w szczególności należy rozumieć **konstrukcję wyrażeń** w Rackecie, potrafić **obliczyć wartość** prostego **wyrażenia** i rozumieć pojęcie **formy specjalnej**. Na zajęciach będziemy się zajmować **podstawieniowym modelem obliczeń** a także **gorliwą i leniwą kolejnością obliczania**¹, należy więc co najmniej zapoznać się z wyjaśnieniami w podręczniku. W celu przyswojenia podstaw przed zajęciami należy wykonać następujące dwa ćwiczenia.

Ćwiczenie 1.²

Przeanalizuj poniższą sekwencję wyrażeń. Jaki wynik wypisze interpreter w odpowiedzi na każde z nich, zakładając że będą obliczane w kolejności w której są podane? Sprawdź swoje przewidywania używając interpretera.

```
10
(+ 5 3 4)
(- 9 1)
(/ 6 2)
(+ (* 2 4) (- 4 6))
(define a 3)
(define b (+ a 1))
(+ a b (* a b))
(= a b)
(if (and (> b a) (< b (* a b)))
```

¹Podręcznik nazywa te strategie obliczania, niezbyt fortunnie, *stosowaną* i *normalną* (ang. *applicative* i *normal evaluation order*).

²Większość ćwiczeń pochodzi z rozdziału 1.1. podręcznika.

```

      b
    a)

(cond [(= a 4) 6]
      [(= b 4) (+ 6 7 a)]
      [else 25])

(+ 2 (if (> b a) b a))

(* (cond [(> a b) a]
         [(< a b) b]
         [else -1])
   (+ a 1))

```

Ćwiczenie 2.

Przedstaw w postaci prefiksowej poniższe wyrażenie:

$$\frac{5 + 4 + (2 - (3 - (6 + \frac{4}{5})))}{3(6 - 2)(2 - 7)}.$$

Na zajęciach

Ćwiczenie 3.

Zastosuj zasady obliczania wyrażeń poznane na wykładzie do obliczenia wartości poniższych wyrażeń. Które z nich spowodują błąd i dlaczego?

```

(* (+ 2 2)      5)

(* (+ 2 2) (5))

(* (+ (2 2) 5))

(* (+ 2
      2) 5)

(5 * 4)

(5 * (2 + 2))

((+ 2 3))

+

(define + (* 2 3))

+

(* 2 +)

```

```
(define (five) 5)

(define four 4)

(five)

four

five

(four)
```

Ćwiczenie 4.

Zdefiniuj procedurę o trzech argumentach będących liczbami, której wynikiem jest suma kwadratów dwóch większych jej argumentów.

Ćwiczenie 5.

Zauważ że w naszym modelu obliczania wartości dopuszczamy, aby operatorami były wyrażenia złożone. Korzystając z tej obserwacji, wyjaśnij działanie następującej procedury:

```
(define (a-plus-abs-b a b)
  ((if (> b 0) + -) a b))
```

Ćwiczenie 6.

Z wykładu wiemy że `and` i `or` są formami specjalnymi, obliczającymi podwyrażenia od lewej do prawej tak długo aż trafi odpowiednio na wartość `false` lub `true`, i nie obliczające pozostałych podwyrażeń. Podaj przykład wyrażenia, którego obliczanie zakończyłoby się błędem *gdyby* `and` był *procedurą wbudowaną*, a nie formą specjalną. Znajdź analogiczny przykład dla formy `or`.

Ćwiczenie 7.

Przeanalizuj poniższe procedury. W jaki sposób możesz użyć ich aby sprawdzić czy interpreter wykonuje obliczenia używając stosowanej czy normalnej kolejności obliczania? Uzasadnij odpowiedź, pokazując jak interpreter wyliczyłby wartość w zależności od kolejności obliczania. Załóż że reguła obliczania wartości formy specjalnej `if` nie zależy od kolejności obliczania.

```
(define (p) (p))

(define (test x y)
```

```
(if (= x 0)
  0
  y))
```

Ćwiczenie 8.

Zdefiniuj procedurę `power-close-to`, która przyjmuje jako argumenty liczby dodatnie b i n , i zwraca najmniejszą liczbę naturalną e taką że $b^e > n$. Możesz użyć wbudowanej procedury `expt` podnoszącej liczbę do danej potęgi.

```
(power-close-to 2 1000)
> 10
```

```
(expt 2 10)
> 1024
```

Użyj struktury blokowej aby ukryć definicje pomocniczych procedur przed użytkownikiem i użyj lokalnego wiązania zmiennych aby usunąć zbędne parametry.

Zadanie domowe (na pracownię)

Metodę Newtona omówioną na wykładzie dla przykładu pierwiastka kwadratowego można zastosować również do obliczania pierwiastka sześciennego. W tym celu wykorzystujemy fakt, że jeśli y jest przybliżoną wartością pierwiastka sześciennego z x , to

$$\frac{\frac{x}{y^2} + 2y}{3}$$

jest lepszym przybliżeniem. Korzystając z tej zależności zaimplementuj procedurę `cube-root`, analogiczną do procedury obliczającej pierwiastki kwadratowe. Pamiętaj aby użyć struktury blokowej i wiązania składniowe żeby uzyskać zwężony kod, i ukryć przed użytkownikiem pomocnicze procedury! Przetestuj też działanie swojej procedury na kilku przykładach lub — jeśli *bardzo* nie lubisz testować — udowodnij jej poprawność.

Uwaga! Plik zawierający definicję funkcji i przykłady testowe w formacie nazwisko-imie.rkt³ należy przesłać w systemie SKOS w *nieprzekraczalnym* terminie **24 lutego 2018 r., godz. 23.55**. Proszę pamiętać o wyszczególnieniu ewentualnych współpracowników w komentarzu.

³Małymi literami, bez spacji i polskich znaków: każdemu z nas przypada *dużo* Waszych prac!