「アルゴリズムとデータ構造入門」
第3回課題

工学部情報学科
平成25年入学
学籍番号：1029-25-2723
森井 崇斗

October 29, 2013

1　繰返型階上のプログラムの作成と実行

1.1　プログラムの内容

```
1  (define (factorial-iter n)
2    (fact-iter 1 1 n)
3    )
4  (define (fact-iter product counter max-count)
5    (if (> counter max-count)
6      product
7      (fact-iter (* counter product)
8                 (+ counter 1)
9                 max-count
10                )
11     )
12   )
```

1.2　出力結果

```
> (factorial-iter 103)
99029007164861804075467152545817733490901658221144924830052805546998766658416222832141441073883538492653516385977292093222882134415149891584000000000000000000000000
```

1.3 説明

1. 初めに引数として n を受け取る。
2. その n を利用して、fact-iter を呼び出す。
3. fact-iter の引数は 3 つあり、counter を max-count （ = n ）に等しくなるまで 1 ずつ大きくしていく。
4. その counter の値を 1 つずつ掛けた物を product に格納する。
5. 3 と 4 を繰り返すことで $1 * 2 * \cdots (n-1)n$ を実現する。

2 教科書 1-2-3〜1-3-1 の想定質問とその回答

2.1 想定質問

1.3.1 で紹介されている sum を利用したプログラムを作成せよ。

参考

```
1 (define (sum term a next b)
2   (if (> a b)
3     0
4     (+ (term a)
5        (sum term (next a) next b))))
```

2.2 想定質問への解答

1〜n までのそれぞれの階乗の和を計算するプログラムを作成する。
愚直に 1〜n までの階乗をそれぞれ求め、それの和を取ることでも求めることが可能であるが、$n!$ は $n(n-1)!$ であることを利用して実装する
$n!$ を求める際に既に $(n-1)!$ の解は求まっているはずであるので、この解を利用すれば計算量を減らすことが可能である。
まず以下に sum をそのまま利用した場合の実装例を示す。

```
1 (define (factorial-iter n)
2   (fact-iter 1 1 n)
3   )
4 (define (fact-iter product counter max-count)
5   (if (> counter max-count)
6     product
7     (fact-iter (* counter product)
8                (+ counter 1)
```

```
 9                 max-count
10                 )
11     )
12   )
13 (define (sum term a next b)
14 (if (> a b)
15   0
16   (+ (term a)
17      (sum term (next a) next b))))
18 (define (inc n) (+ n 1))
19     (define (sum-facts a b)
20        (sum factorial-smart a inc b))
```

次に前述の方法で計算量を減らした場合の実装例を示す。

```
 1 (define (fact-sum-smart n)
 2  (define (iter counter product tmp)
 3  (if (< n counter)
 4   product
 5   (iter (+ counter 1)
 6         (+ product (* tmp (+ counter 1)))
 7         (* tmp (+ counter 1))
 8          )
 9  )
10  )
11  (iter 1 1 1)
12  )
```