CS入門　課題2　演習ガイド

本日の予定

1. 準備
2. ango.rb, hukugo.rb の作成
3. kaidoku.rb のアイデア
4. kaidoku.rb の作成など

1. 準備

1. ログインする.
2. Terminal を動かす（TSUBAME と直接対話する窓口）.
   - 2.1. mkdir kadai3　課題2の部屋（フォルダ）を作る.
   - 2.2. 必要なファイルを共通のお部屋から kadai3 へ
     コピーする.

共通ファイルの置き場所：Desktop/shared/CS/cs1/kadai3

2. 暗号化，復号プログラムの作成

ango.rb, hukugo.rb の作成

1. まずは，コピーしてきた復習用の code.rb を実行してみよう.

```
code_a = 97              # 文字 a の文字コード
kosu = 26                # 英字アルファベットの数

bun = gets.chomp         # 入力文字列から改行を除去
cc = bun.unpack("C*")    # 文字列懼・文字コードの配列
leng = bun.length        # 文字列の長さ

for i in 0..leng-1
  moji = bun[i]          # bun の i 文字目を得る (i は 0から始まる)
  code = cc[i]           # その文字のコードを得る
  sa = code - code_a  # 文字 a との差分
  if  0 <= sa && sa < kosu                    # 小文字アルファベットなら
     print(moji, ": ", code, ", ", sa, "¥n")  #      差分まで表示する
  else                                        # そうでないときは
     print(moji, ":", code,  "¥n")            #      差分は表示しない
  end
end
```

2. これを参考に，ango.rb, hukugo.rb を完成させよう.

2. 暗号化，復号プログラムの作成

ango.rb, hukugo.rb の作成

2. これを参考に，ango.rb, hukugo.rb を完成させよう.

```
# enc(秘密鍵 k, 平文 m) = 暗号文 c
def enc(k, m)
  code_a = 97            # 文字 a の文字コード
  kosu = 26              # 英字アルファベットの数
  leng = m.length        # 文字列の長さ
  a = m.unpack("C*")     # 文字列から文字コードの配列へ変換
  b = Array.new(leng)    # 暗号文（のコード）格納用配列
  for i in 0..leng-1
      code = a[i]              # i 文字目のコードを得る
      sa = code - code_a       # 文字 a からの差分

      b[i] =
  end
  c = b.pack("C*")         # コードの配列を文字列に直す
  return(c)
end
# サブルーチン enc（終）
```

ここを作る

使い方

```
k = 3
hirabun = gets.chomp          # 平文を入力
angobun = enc(k, hirabun)     # 暗号文に変換
puts(angobun)                 # 暗号文を出力
```

2. 暗号化，復号プログラムの作成

3. 作った ango.rb, hukugo.rb の使い方

```
$ ruby ango.rb
  Hello, love you!
  Hhoor, oryh brx!
$
```

m.txt

Hello, love you!

前もって安全なところで作っておく

Terminal 上での使い方

```
$ ruby ango.rb < m.txt
  Hhoor, oryh brx!
$
```

- 入力データをファイルから読み込む
  ruby ango.rb < ファイル名
- 出力をファイルに書き出す
  ruby hukugo.rb > ファイル名

※ 読み込んで書き出すことも可能

ruby ango.rb < hirabun.txt > angobun.txt

3. 解読プログラムのアイデア

解読 ⟷ 秘密鍵を知らない者が暗号文から平文を得ること

比較的長い英文を暗号化したものを解読したい
どうすればよいか？

明らかだよ
ワトソン君

宿題：考えてきて下さい

英語の場合

一番多く現れる文字が e のはず！

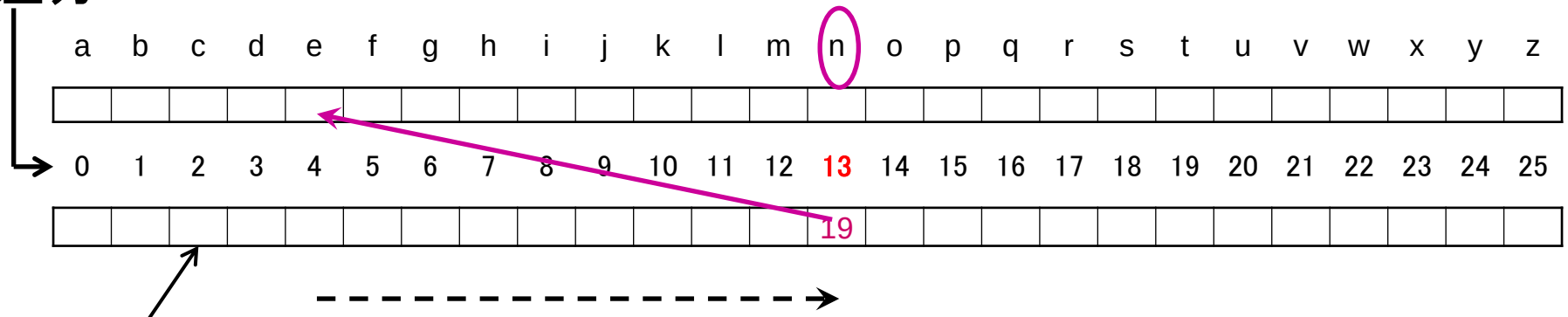
qxuv**n**b qjm k**nn**w b**n**jc**n**m oxa bxv**n** qxdab rw bru**n**wl**n** frcq qrb
uxwp, cqrw kjlt ldae**n**m xe**n**a j lqnvrlju e**n**bb**n**u rw fqrlq q**n**
fjb ka**n**frwp j yjacrldujauh vjuxmxaxdb yaxmdlc. qrb q**n**jm
fjb bdwt dyxw qrb ka**n**jbc, jwm q**n** uxxt**n**m oaxv vh yxrwc xo ...

n が19回出現で最多

qxuv**n**b qjm k**nn**w b**n**jc**n**m oxa bxv**n** qxdab rw bru**n**wl**n** frcq qrb
uxwp, cqrw kjlt ldae**n**m xe**n**a j lqnvrlju e**n**bb**n**u rw fqrlq q**n**
fjb ka**n**frwp j yjacrldujauh vjuxmxaxdb yaxmdlc. qrb q**n**jm
fjb bdwt dyxw qrb ka**n**jbc, jwm q**n** uxxt**n**m oaxv vh yxrwc xo ...

差分

n が19回出現で最多



13−4 = 9 だけずれた ⇒ k = 9

頻度配列と呼ぼう

アイデア

1. 頻度配列 hindo を作る.
2. 最大頻度の場所 maxj を見つける.
3. k = maxj − 4 で求め, dec(k, angobun) で平文を求める.

注意！ maxj < 4 のときも
大丈夫！？

まとめ Terminal 上のコマンド

命令	使用例	意味
mkdir	mkdir kadai2	kadai2 というフォルダ（部屋）を作る
cd	cd kadai2	kadai2 というお部屋に入る
	cd ..	上の（大きな）部屋に戻る
	cd ../..	上の上の部屋に戻る
ls	ls	その部屋にあるファイルを表示する
rm	rm foo.rb	foo.rb を消す（戻らないので注意）
リダイレクト <	ruby xx.rb < aa	xx.rb を実行．入力は aa から取り込む
リダイレクト >	ruby xx.rb > bb	xx.rb を実行．結果は bb へ出す

まとめ（復習） Ruby での書き方（その1）

【演算子】

演算	使用例	意味
+	x + y	x と y の足し算
-	x - y	x から y の引き算
*	x * y	x と y の掛け算
/	x / y	x を y で割った商
%	x % y	x を y で割った余り
**	x ** y	x の y 乗

【関係演算子】

関係	使用例	意味
>=	x >= y	x は y より大きいかまたは等しい
>	x > y	x は y より大きい
==	x == y	x は y と等しい
!=	x != y	x は y は等しくない
<	x < y	x は y より小さい
<=	x <= y	x は y より小さいかまたは等しい

まとめ（復習） Ruby での書き方（その2）

【論理演算子】

論理記号	使用例	意味
&&	x && y	x と y の論理積 (両方が真のとき真)
||	x || y	x と y の論理和 (少なくとも一方が真のとき真)
!	!x	x の否定 (x が真のとき偽，x が偽のとき真)

まとめ（復習） Ruby での書き方（その3）

【配列】

初期設定　aa = [0, 0, -5, 4]

aa = Array.new(4)　← 要素数 4 の配列生成し aa とする

aa = Array.new(4, 0)　← 各要素の初期値が 0

指定方法　aa[*i*]　＝ aa の *i* 番目（添え字 *i* は 0 から）

コマンド　aa.length　＝　配列 aa の長さ（＝要素数）

※「添え字」は「インデックス」(index) ともいう.

【文字列】

初期設定　s = "Coffee+milk"

指定方法　s[*i*]　＝　s の *i* 文字目（添え字 *i* は 0 から）

コマンド　s.length　＝　文字列列 s の長さ

a = s.unpack("C*")　← s の各文字を ASCII に直して
配列 a に格納する

s = a.pack("C*")　← 配列 a の各数字を文字に直して
文字列用変数 s に格納する

まとめ（復習） Ruby での書き方（その4）

【繰り返し文】

```
while 条件式
    ・・・
end
```

← 条件式の成立している間
・・・ を繰り返す

```
for  m in a..b
    ・・・
end
```

← 変数 m の値を a から b まで
1 ずつ増加させながら ・・・ を繰り返す

【条件分岐文】

```
if  条件式
    ・・(A) ・・
else
    ・・(B)・・
end
```

← 条件式の成立したときは ・・(A) ・・を実行

← そうでないときは ・・(B) ・・を実行

省略可

まとめ Ruby での書き方（その5）

【入出力】

puts(a, b, "hello", c)　← 変数 a, b の値，文字列 hello，変数 c の値を改行しながら画面に表示する

print(a, b, "hello", c, "¥n")　← 上と同様．ただし改行はしない．空白も空けない．従って，最後には改行記号を画面に出すことで改行させる．

※ ¥ は「バックスラッシュ」という記号を表わしている．改行を意味する記号として用いられることが多い．
Mac では Option キーと ¥ キーを同時に押すとタイプできる．