I117（21）テキストファイル中の単語頻度

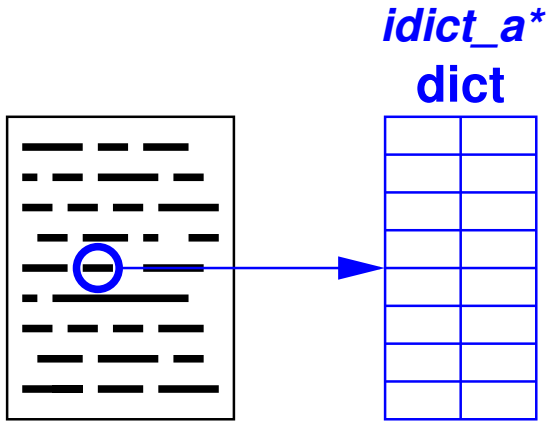
知念

北陸先端科学技術大学院大学 情報科学研究科
School of Information Science,
Japan Advanced Institute of Science and Technology

テキストファイル中の単語頻度

テキストファイルから単語を拾って頻度を集計する

```
while( 1文字読みとる ) {
    if( 単語を構成する文字 ) {
        単語を作る
    }
    if( 単語ができたら ) {
        辞書に登録と頻度計上
    }
}
```



※ 以前、1行1単語の場合の頻度は扱った

単語切りだし

```
int wordbreak(idict_a *dict){
  char  word[BUFSIZ];
  char *q;  int ch, c;
  q = word;  c = 0;
  do {
    ch = fgetc(stdin);
    if(isalnum(ch)||ch=='-'||ch=='.'||ch=='_'){
       if(c<BUFSIZ) { *q++ = ch%256; c++; } }
    else {
      *q = '\0';
      if(c>0) {  store(dict, word);  }
      q = word;  c = 0;  }
```

単語切りだし *(cont.)*

```
  } while(ch!=EOF);
  if(c>0) {
    *q = '\0';
    store(dict, word);
  }
}
int main() {
  idict_a *sd;
  sd = idict_new();  idict_init(sd);
  wordbreak(sd);
  idict_sortbyvalue(sd);  idict_print(sd);
}
```

単語切りだし *(cont.)*

- 文字単位処理
- 英数字、ハイフン、アンダースコア、ドットを単語構成要素とした
- 標準ライブラリ関数 isalnum(int x) は文字xが英数字の際、1 を返す
- idict_sortbyvalue() は idict_c のメンバ value にしたがって整列する

格納

```
int store(idict_a *dict, char *word) {
    idict_c *pos;
    pos = idict_findpos(dict, word);
    if(pos) {  pos->value++;  }
    else {     idict_add(dict, word, 1);
    return 0;
}
```

恒例の処理

- 格納されていない場合、初期値１で格納
- 格納されているとカウンタの値を1つ増やす

コンパイルと実行

```
% cc -o wb01 wb01.c libidict.c
wb01.c:
libidict.c:
```

加工データ

```
% ls -l *.c
-rw-r--r-- 1 chinen is    2716 Jul  8 18:15 libidict.c
-rw-r--r-- 1 chinen is    2481 Jul  4 15:06 libsdict.c
-rw-r--r-- 1 chinen is    1079 Jul  8 19:07 wb01.c
-rw-r--r-- 1 chinen is    1488 Jul  8 19:08 wb02.c
-rw-r--r-- 1 chinen is    1680 Jul  8 19:03 wb02b.c
-rw-r--r-- 1 chinen is    1764 Jul  8 19:21 wb03.c
```

コンパイルと実行 *(cont.)*

実行結果

```
% ls -l *.c | ./wb01 | pr -t -3
#idict expand 10 -> 20
#idict expand 20 -> 40
 0 06 1             9 03 1           18 21 1
 1 2481 1          10 1764 1         19 15 2
 2 2716 1          11 07 1           20 19 4
 3 libidict.c 1    12 08 1           21 8 5
 4 libsdict.c 1    13 1079 1         22 is 6
 5 wb01.c 1        14 1488 1         23 chinen 6
 6 wb02.c 1        15 1680 1         24 -rw-r--r-- 6
 7 wb02b.c 1       16 4 1            25 Jul 6
 8 wb03.c 1        17 18 1           26 1 6
```

バックスペース除去

バックスペースを使って文字を強調する手法がある

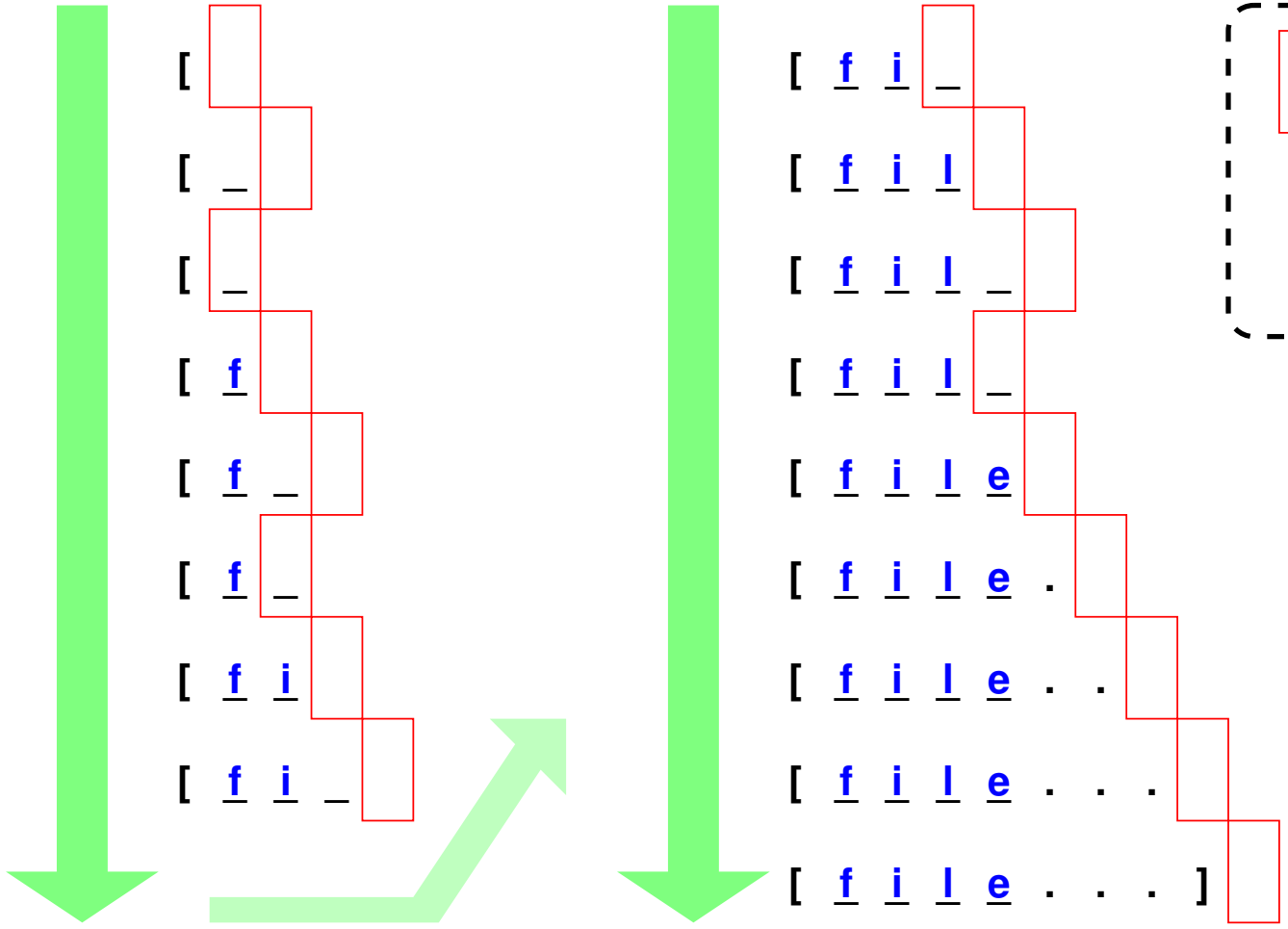
- 後退して重ね打ちするタイプライタ由来の手法

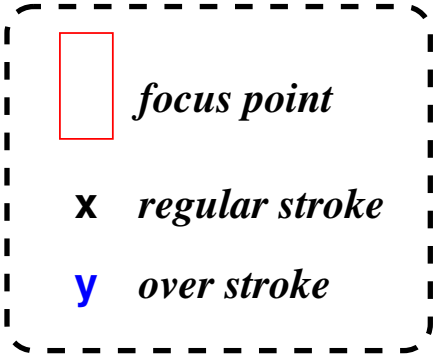
```
NAME
     ls - list contents of directory

SYNOPSIS
     /usr/bin/ls [-aAbcCdfFghilLmnopqrRstux1@] [file...]

     /usr/xpg4/bin/ls [-aAbcCdfFghilLmnopqrRstux1@] [file...]

DESCRIPTION
     For each file that is a directory, ls lists the contents
     the  directory.  For  each file that is an ordinary file,
```

今回はアンダースコアとバックスペースで下線を作る

```
0000000                  /  u  s  r  /  x  p  g  4  /  b
0000020  i  n  /  l  s     [  -  a  A  b  c  C  d  f  F
0000040  g  h  i  l  L  m  n  o  p  q  r  R  s  t  u  x
0000060  1  @  ]     [  _ \b  f  _ \b  i  _ \b  l  _ \b
0000100  e  .  .  .  ] \n \n  D  E  S  C  R  I  P  T  I
0000120  O  N \n                 F  o  r     e  a  c  h
0000140     _ \b  f  _ \b  i  _ \b  l  _ \b  e     t  h
0000160  a  t     i  s     a     d  i  r  e  c  t  o  r
0000200  y  ,     l  s     l  i  s  t  s     t  h  e
0000220  c  o  n  t  e  n  t  s        o  f \n
0000240        t  h  e        d  i  r  e  c  t  o  r  y
0000260  .        F  o  r        e  a  c  h     _ \b  f
0000300  _ \b  i  _ \b  l  _ \b  e     t  h  a  t     i
0000320  s     a  n     o  r  d  i  n  a  r  y     f  i
0000340  l  e  ,     l  s \n
```

出力の流れ — 重ね打ちになっている





```
% man ls | head -18 | tail -5 | ./wb01 | pr -t -w 50
(略)
 0 -aAbcCdfFghil  9 file 1          18 is 2
 1 DESCRIPTION 1 10 lists 1         19 that 2
 2 a 1           11 of 1            20 the 2
 3 an 1          12 ordinary 1      21 _ 3
 4 bin 1         13 usr 1           22 f_ 3
 5 contents 1    14 xpg4 1          23 i_ 3
 6 directory 1   15 For 2           24 l_ 3
 7 directory. 1  16 e 2             25 ls 3
 8 e... 1        17 each 2
```

案の定、_, f_, i_, l_, e という項目が計上されている。

バックスペース除去 *(cont.)*

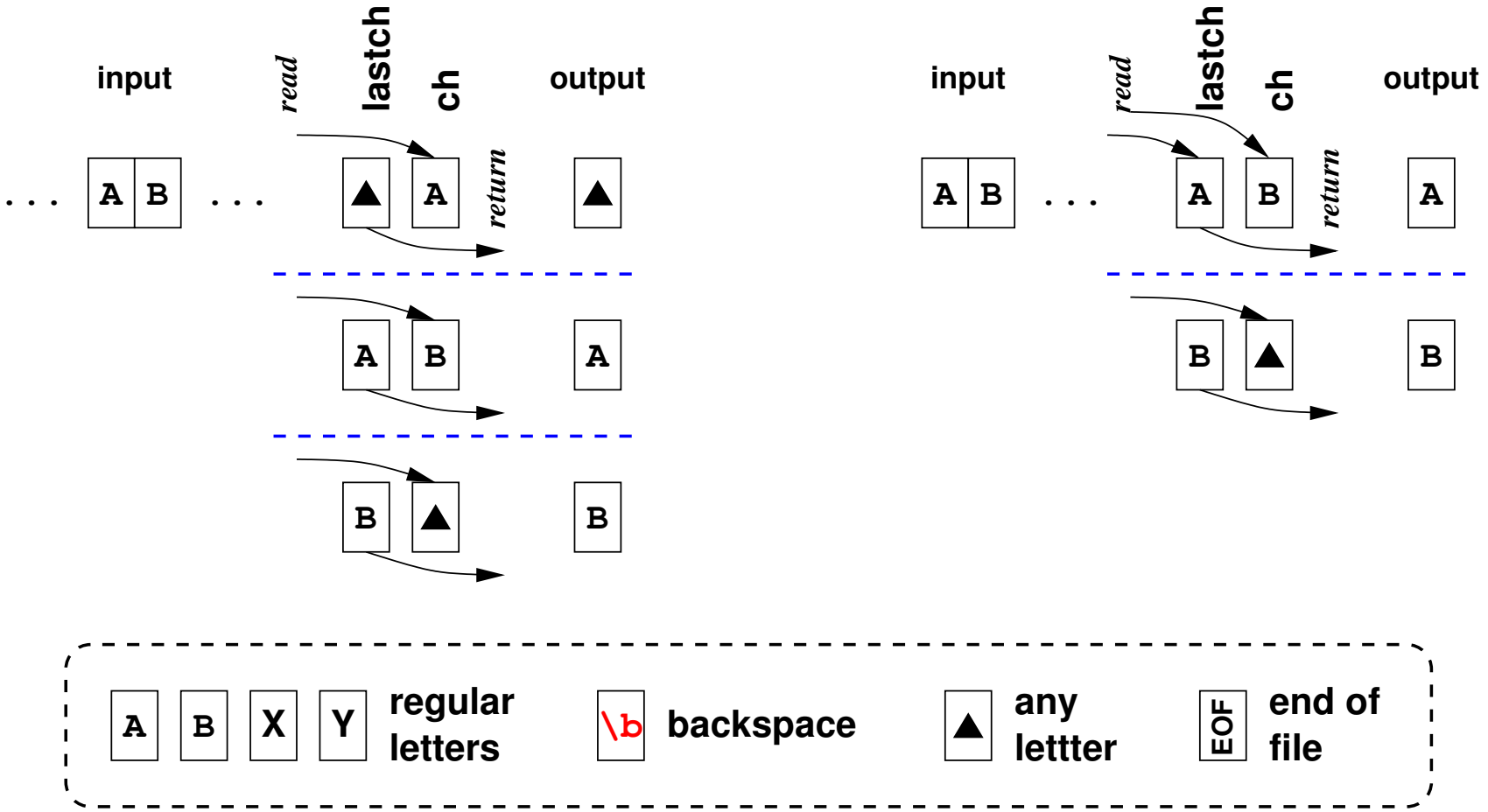
- 後退するために過去の文字を保持する
- 幸い1文字なので、安直に実装可能
  - fgetc() の代理に bfgetc() を作る
    - 最初だけ 2文字を読み込む
    - それ以外は 1文字読み込む
    - 読み込んだ文字がバックスペース(\b 0x08) なら 2文字とも捨ててやり直し
    - 先に読んだ方を返す
  - 残りは同じ

```
#define IG  (-23)
int bfgetc(FILE *fp){
    static int lastch=IG; int ch, r;
    if(lastch==IG) {
first:  lastch = fgetc(stdin);
    }
    ch = fgetc(stdin);
    if(ch=='\b') { goto first; }
    r = lastch;
    lastch = ch;
    return r;
}
```

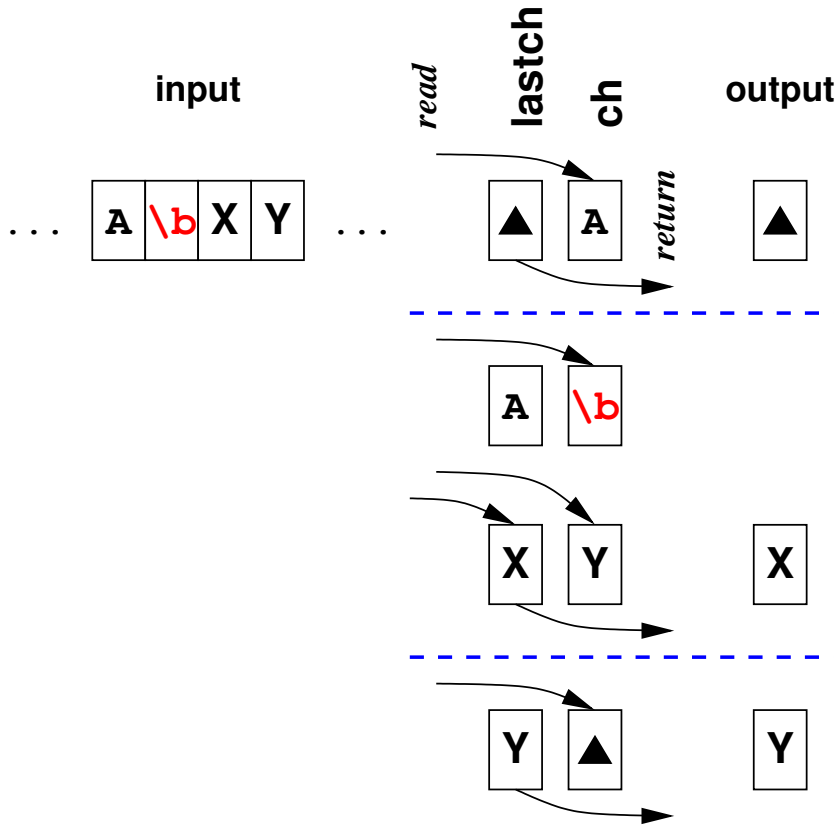
バックスペース除去 *(cont.)*

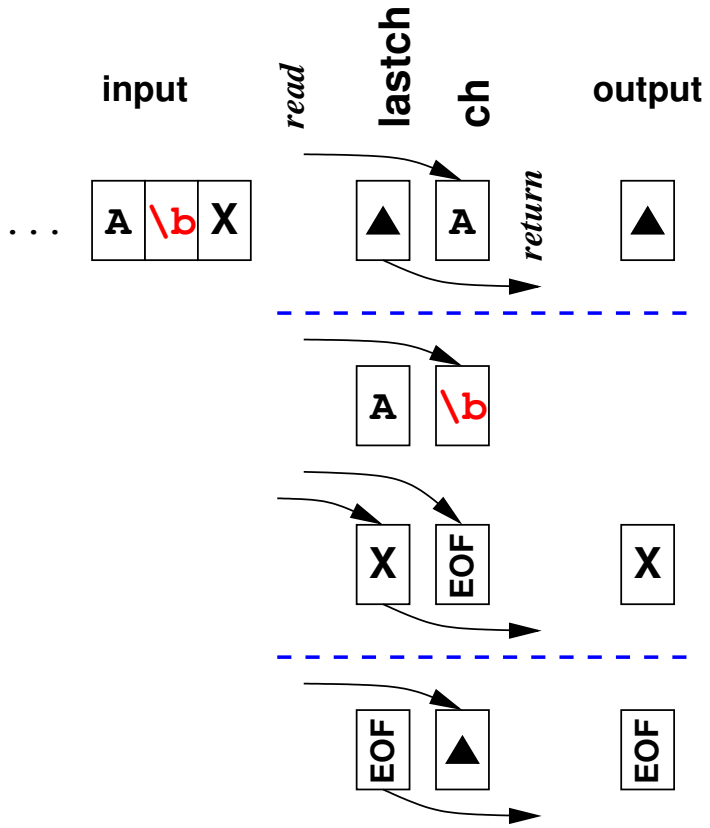
- IG は初期値
  - EOF (値 -1) とぶつからないために -23 とした
  - static 変数なので値はずっと保持される
  - 関数が呼びし毎の初期化はない
- 最初は lastch が IG なので判定できる
  - 2 文字体制にするため、1文字余計に読み込む
- \b が見付かると 2文字読むためにラベル first にジャンプ
- lastch を返す

バックスペースを含まない場合



バックスペースを含む場合





```
% man ls | head -18 | tail -5 | ./wb02 | pr -t -w 50
Reformatting page.  Please Wait... done
#idict expand 10 -> 20
#idict expand 20 -> 40
 0 -aAbcCdfFghil  7 directory. 1  14 For 2
 1 DESCRIPTION 1  8 file... 1     15 each 2
 2 a 1            9 lists 1       16 is 2
 3 an 1          10 of 1          17 that 2
 4 bin 1         11 ordinary 1    18 the 2
 5 contents 1    12 usr 1         19 file 3
 6 directory 1   13 xpg4 1        20 ls 3
```

file や file... を正しく認識した

日本語

- 日本語の単語切りだしは難しい
- この講義の範疇を越える
  - 字種（仮名、漢字）で判定する技法
  - 辞書で判定する技法
  - 興味がある者は「形態素解析」で検索せよ
- ここでは軽くツールを使うことにする
  - 茶筌 ChaSen を用いる
    - 実行ファイル名は chasen

二人はあたりを眺めながら、青田の間を歩いて行った。するとたちまち道ばたに農夫の子らしい童児が一人、円い石を枕にしたまま、すやすや寝ているのを発見した。加藤清正は笠の下から、じっとその童児へ目を落した。

『金将軍』芥川龍之介

この文字列を chasen -j に与えると

```
(略)
加藤	カトウ	加藤	名詞-固有名詞-人名-姓
清正	キヨマサ	清正	名詞-固有名詞-人名-名
は	ハ	は	助詞-係助詞
笠	カサ	笠	名詞-一般
の	ノ	の	助詞-連体化
下	シタ	下	名詞-一般
から	カラ	から	助詞-格助詞-一般
、	、	、	記号-読点
じっと	ジット	じっと	副詞-一般
その	ソノ	その	連体詞
童	ワラベ	童	名詞-一般
児	ジ	児	名詞-接尾-一般
```

日本語 *(cont.)*

```
へ      へ      へ      助詞-格助詞-一般
目      メ      目      名詞-一般
を      ヲ      を      助詞-格助詞-一般
落し    オトシ  落す    動詞-自立       五段・サ行      連用形
た      タ      た      助動詞  特殊・タ        基本形
。      。      。      記号-句点
EOS
```

- 形態素に分解した結果を出力する
- 行が終ると EOS という文字列が入る
- 行単位の処理とした方が素直に実装できる

ChaSen 結果の利用

```
int wordbreak(idict_a *dict) {
  char line[BUFSIZ];  char *p;
  while(fgets(line, BUFSIZ, stdin)) {
    p = line;
    if(*p=='E' && *(p+1)=='O' && *(p+2)=='S') {
        continue;
    }
    while(*p) {
      if(*p=='\t') { *p = '\0';  break;  } p++; }
    store(dict, line);
  }
}
```

```
% nkf -e k.txt | chasen -j | ./wb03 | tail -48 | pr -t -w 38 -4
(略)
268 お 2          280 清正 4        292 ― 5    304 ない 9
269 朝鮮 3        281 から 4        293 … 5    305 と 11
270 枕 3          282 児 4          294 その 5    306 て 11
271 何 3          283 童 4          295 か 5    307 で 12
272 日本 3        284 二 4          296 人 6    308 、 18
273 よう 3        285 この 4        297 」 6    309    20
274 香 3          286 い 4          298 金 6    310 に 31
275 もの 3        287 将軍 4        299 が 6    311 を 32
276 国 3          288 も 5          300 「 7    312 た 35
277 石 3          289 歴史 5        301 へ 8    313 。 35
278 小倅 3        290 いる 5        302 し 8    314 は 35
279 小西 3        291 行長 5        303 ある 8    315 の 47
```

清正、行長、小西などの形態素が取り出せている

補足

- 重ね打ちの技法
  - 同じ文字を二つ打って強調する(太字 bold 相当)
  - X や ＊ 等を重ねて打って否定する
- 日本語文章の特徴抽出では、品詞を考慮して処理する(名詞のみ抜き出す等)ことが多い
  - 今回のような単純な例だと助詞や助動詞が大量に出てくる

演習

1) 単語頻度計上プログラムを変更して、のべ単語数を算出するプログラムを作れ★
2) バックスペースを考慮した単語頻度計上プログラムの bfgetc() を goto を使わず実現せよ★
3) バックスペースを考慮した単語頻度計上プログラムで ungetc() を用いるようプログラムを変更せよ★
4) 英文のファイルから行区切りではなく、ピリオドで論理的な行を取り出すプログラムを作れ★

演習 *(cont.)*

5) 和文のファイルから行区切りではなく、読点で論理的な行を取り出すプログラムを作れ★★

6) ChaSen の出力する品詞情報を参考に、登場する名詞の頻度を算出するプログラムを作れ★★