

コンパイラ及び演習

関澤 俊弦

日本大学 工学部 情報工学科

連絡

■ 演習課題提出システム

- ユーザ登録を行っていない場合、登録を行ってください
 - 参考: 第1回講義資料
 - Registration Codeは関澤に聞いてください

■ 原則として課題の提出期間は延長しません

- 前回までの講義内容を利用し、講義毎に内容を加えていくため

補足：演習課題提出システム

■ 100点でない場合の差異



The screenshot shows a submission system interface with the following elements and annotations:

- Submitted File Name:** `compiler02_step3_wrong.c` (Annotated with a red box and a line pointing to the text "提出したファイル名").
- Submit ID:** 424
- Username:** (Empty)
- Problem:** 3
- Test 1:** WRONG (Annotated with a red box and a line pointing to the text "テスト番号と結果 (ACCEPT / WRONG)").
- Submitted Output:**

```
< a (n) b (n) c (y) d (y) e (y) f (y) g (y)
---
```

(Annotated with a red box and a line pointing to the text "提出ファイルの出力 ' < ' で始まる行").
- Expected Output:**

```
> a(Y) b(Y) c(N) d(N) e(N) f(N) g(N)
```

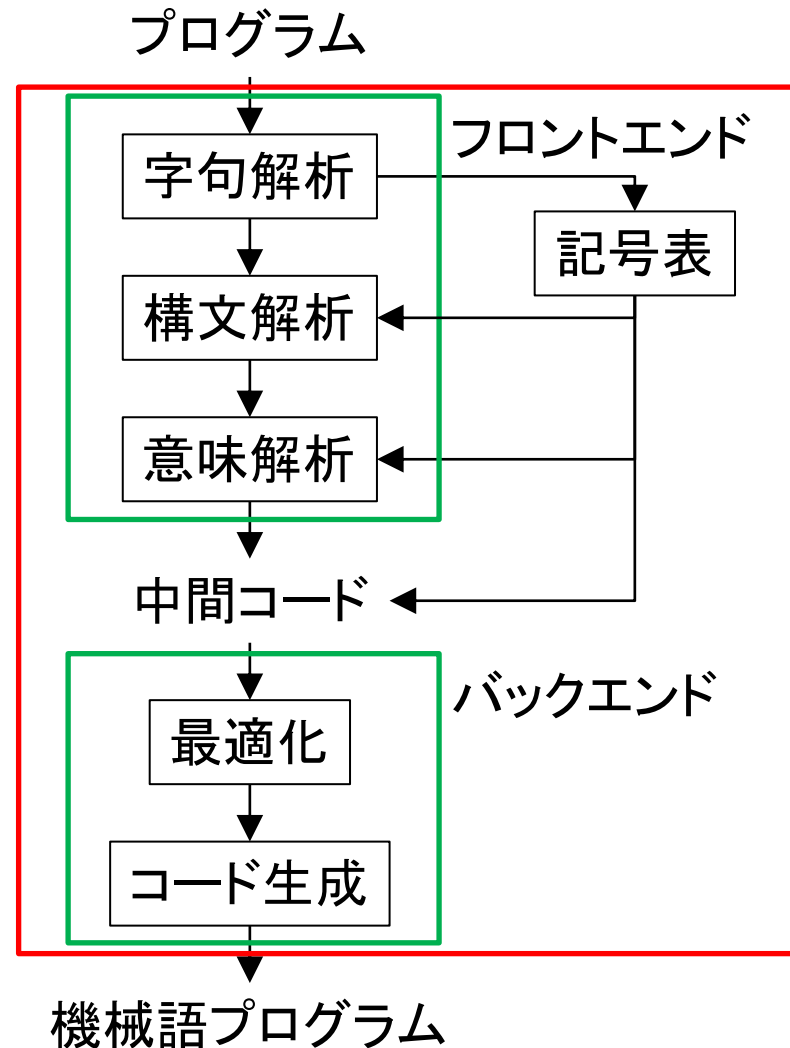
(Annotated with a red box and a line pointing to the text "期待される出力 '>' で始まる行").
- Test 2:** WRONG
- Submitted Output for Test 2:**

```
3c3
< x (y) y (y) z (y)
---
```
- Expected Output for Test 2:**

```
> x(N) y(N) z(N)
```

復習

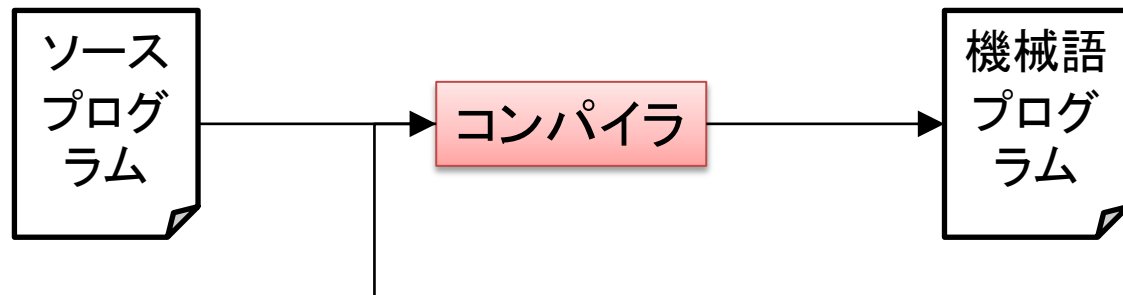
- 言語処理系
- コンパイラとは
 - 処理過程
 - 動作概要
 - T図式



復習

■ コマンドラインからのファイル名の指定

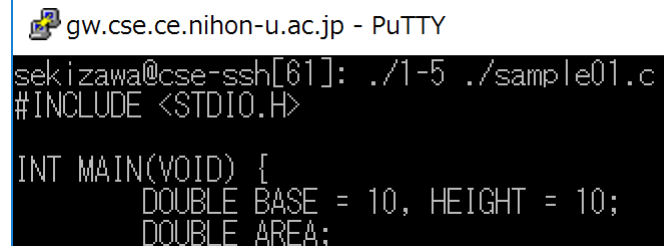
□ コンパイラはソースプログラムを処理



```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[]) {
    ...
    exit(EXIT_SUCCESS);
}
```

コマンドラインでの
ファイル名の指定



```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh[61]: ./1-5 ./sample01.c
#include <STDIO.H>

INT MAIN(VOID) {
    DOUBLE BASE = 10, HEIGHT = 10;
    DOUBLE AREA;
```

- 記号, 語, 言語
- 形式言語
 - アルファベット, スター閉包 Σ^* , Σ 上の言語
- プログラミング技法

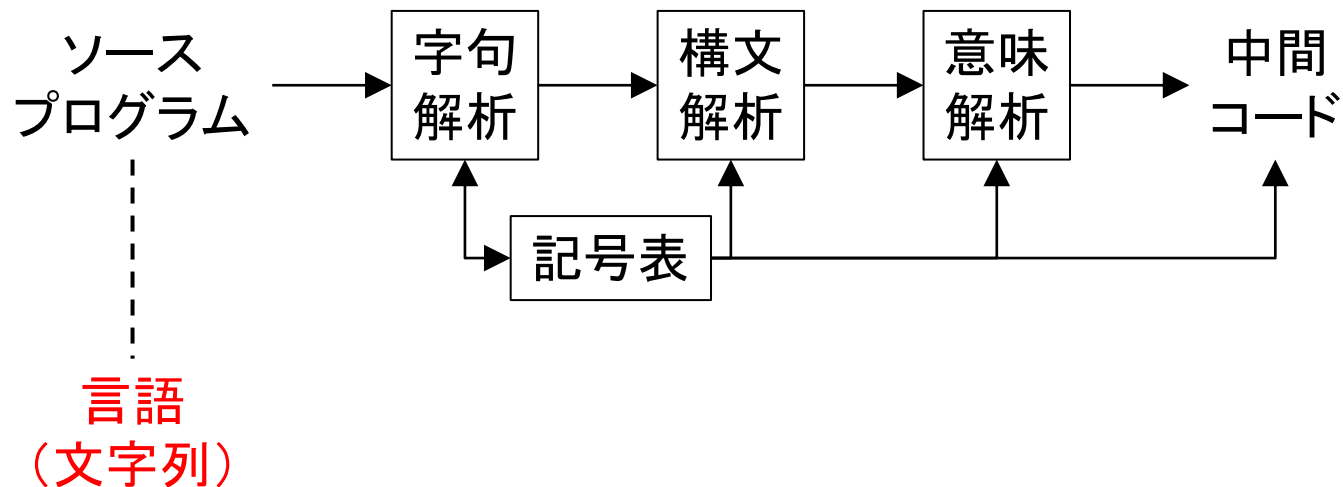
- 記号, 語, 言語
- 形式言語
 - アルファベット, スター閉包 Σ^* , Σ 上の言語
- プログラミング技法

コンパイラ: フロントエンド

■ ソースプログラム

□ CやJavaなどのプログラム言語で記述

- プログラムは文字列
- コンパイラから見ると意味を持った文(文字列)



記号, 語, 言語

■ 記号とは

- 一定の事柄を指し示すために用いる知覚の対象物. 文字などが代表的なもの. (広辞苑第6版)

■ 語とは

- 記号を並べたもの

- 例: 記号として a, b が使えるとき, $a, aaa, abbaba$ など

■ 言語とは

- ある基準を満たしている語を集めたもの

- 基準が「 a の数が奇数個」のとき, $a, aaa, aaaaaa$ など
- 基準が「英語である」のとき, $Betty\ is\ a\ pretty\ girl.$ など

言語の種類

■ 自然言語

□ 人間が意思疎通するために使用する言語

- 日本語, 英語, ...
- 英字で構成されるアルファベット上の記号列の集合

■ 形式言語

□ 文法や意味が形式的に与えられる言語

- プログラミング言語, ...
- アスキー記号(文字)の集合上の記号列の集合

言語の規定方法

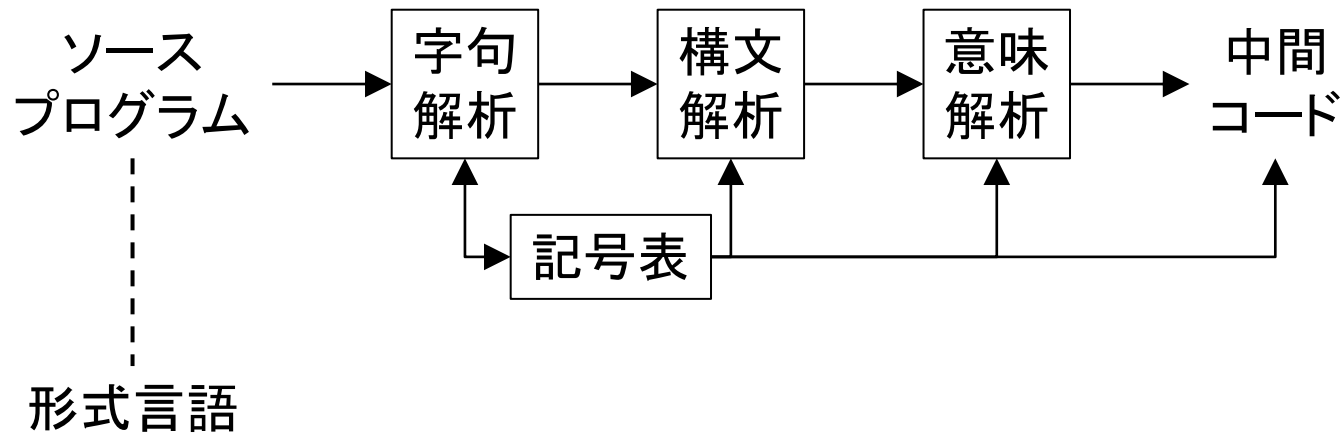
- 言語に属するすべての語を生成する手続きを与える
 - 語の生成装置(文法)
- 任意の語が与えられたとき, その語が言語に属するかどうかを判定する手続きを与える
 - 語の認識装置(オートマトン)

- 記号, 語, 言語
- 形式言語
 - アルファベット, スター閉包 Σ^* , Σ 上の言語
- プログラミング技法

形式言語

■ 自然言語やプログラミング言語などを抽象化して定められた言語

- アルファベット (記号の集合)
- 記号列 (文字列)



アルファベット

■ アルファベット Σ

- 記号の空でない有限集合. 言語を構成する最小単位の領域を定める.
- 例
 - $\Sigma = \{0, 1\}$: 2進数のアルファベット,
 - $\Sigma = \{a, b, c, \dots, z\}$: 小文字の集合,
 - Σ : アスキー記号の集合, など.

記号列 (文, 語, 文字列)

形式的に $\Sigma = \{a_1, a_2, a_3, \dots, a_m\}$ としたとき,

■ Σ 上の記号列 w

□ Σ の要素 (記号) の重複を許した有限列

$$w = a_{i_1} a_{i_2} \dots a_{i_n}$$

- w の長さ $|w|$

w を構成する記号の個数 n で定義される.

- 空記号列 ε

長さが 0 の記号列. 空列, 空系列ともいう.

□ 例

- 2進数のアルファベット $\Sigma = \{0, 1\}$ に対して, 01011 や 111 は文字列

記号列の連接

■ (記号列の) 連接

□ 記号列を繋げてできる記号列

- 記号列 $a = a_0 a_1 \dots a_n$, $b = b_0 b_1 \dots b_m$ のとき,
連接 $ab = a_0 a_1 \dots a_n b_0 b_1 \dots b_m$

□ 記号列の基本的演算

- 空記号列 ε は連接に対する単位元.
すなわち, 任意の記号列 w に対して, $\varepsilon w = w = w\varepsilon$



単位元とは, 他の元 x に対して演算を行なっても, x が変化しない元.

加法の単位元は0, 乗法の単位元は1.

記号列の表記

■ 同じ記号が続くときの表記

□ $a \in \Sigma$ として,

a が n 個並んだ記号列 $\underbrace{aaa \dots a}_{n\text{個}}$ を a^n と書く

- $a^0 = \varepsilon$

- $a^1 = a$

- $a^2 = aa$

- ...

- $a^n = \underbrace{aa \dots a}_{n\text{個}}$

アルファベットのベキ

■ 長さ k の記号列の集合 Σ^k

□ Σ から作られる長さ k の記号列の集合

□ 例

- 任意の Σ に対して, $\Sigma^0 = \{\varepsilon\}$
(長さ0の記号列は ε のみ)

- $\Sigma = \{0, 1\}$ に対して, $\Sigma^1 = \{0, 1\}$, $\Sigma^2 = \{00, 01, 10, 11\}$

■ 空でない記号列の集合 Σ^+

□ $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \dots$

スター閉包

■ スター閉包 Σ^*

□ $\Sigma^* = \Sigma^+ \cup \{\varepsilon\}$

空記号列 ε を含め, Σ 上で作りうるすべての記号列の全体から成る無限集合

□ 例

- $\Sigma = \{0, 1\}$ のとき,
 $\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, \dots\}$

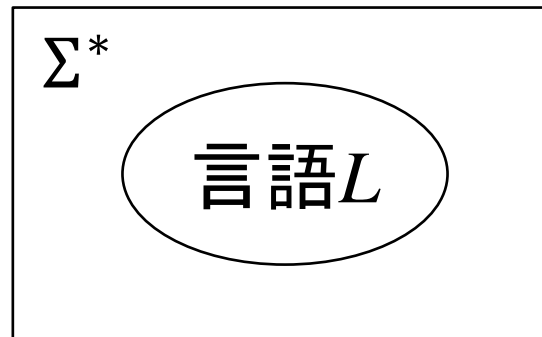
Σ 上の言語

■ Σ 上の言語 L

□ Σ^* の要素の中から, 特定の条件を満たす記号列の集合

- Σ^* の部分集合

- 形式言語における言語となるため, Σ 上の言語と呼ばれる.
- 一般に, L は無限集合



言語の例

■ 英語

- すべての英字で構成されるアルファベット Σ 上の記号列の集合

■ プログラミング言語

- アスキー記号(文字)の集合上の記号列の集合

■ 理論的な言語

- n 個の0の後に n 個の1が並ぶ言語
 - $\{\varepsilon, 01, 0011, 000111, \dots\}$
- 素数を表わす2進数
 - $\{10, 11, 101, 111, 1011, \dots\}$

- 記号, 語, 言語
- 形式言語
 - アルファベット, スター閉包 Σ^* , Σ 上の言語
- プログラミング技法

stdbool.h

■ 真理値型や真偽を定義する

- C99で導入された

■ 機能

- bool型（正確には _Bool型）
- true, false（それぞれ真, 偽に対応）

■ 例

- 戻り値としてbool型のtrueを返す関数func

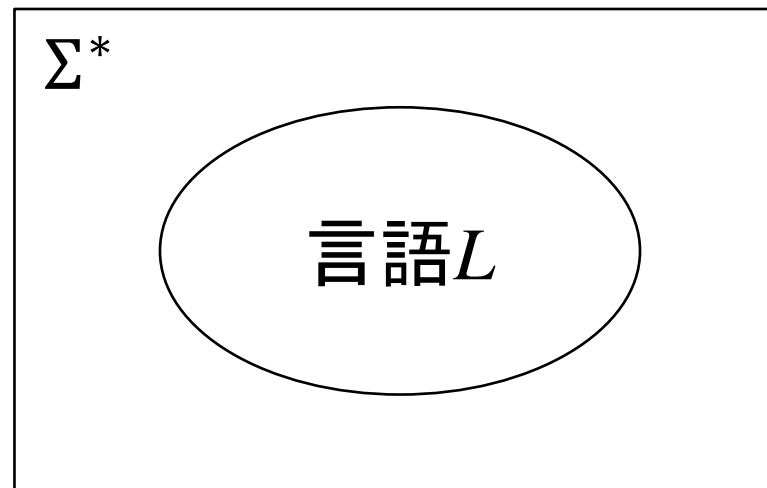
```
#include <stdbool.h>

bool func(); //プロトタイプ宣言

bool func() {
    :
    return true;
}
```

まとめ

- 記号, 語, 言語
- 形式言語
 - アルファベット, スター閉包 Σ^* , Σ 上の言語



演習

演習2-1: Σ 上の記号列

アルファベット $\Sigma = \{a, b\}$ として, Σ 上の記号列 w を考える.

1. Σ^0 を示せ
2. Σ^1 をすべて示せ
3. Σ^2 をすべて示せ
4. Σ^* に含まれる要素を5つ示せ
5. Σ^* に含まれない要素を5つ示せ

演習2-2: 言語判定機の実装

■ 目的

- $\Sigma = \{a, b\}$ としたとき, 入力された記号列 w が Σ^* に含まれるか否か判定する判定機を実装する.
 - 参考: 演習2-1

■ 仕様

- 記号列 w の入力は標準入力とする
- $|w|$ の最大値は255とする

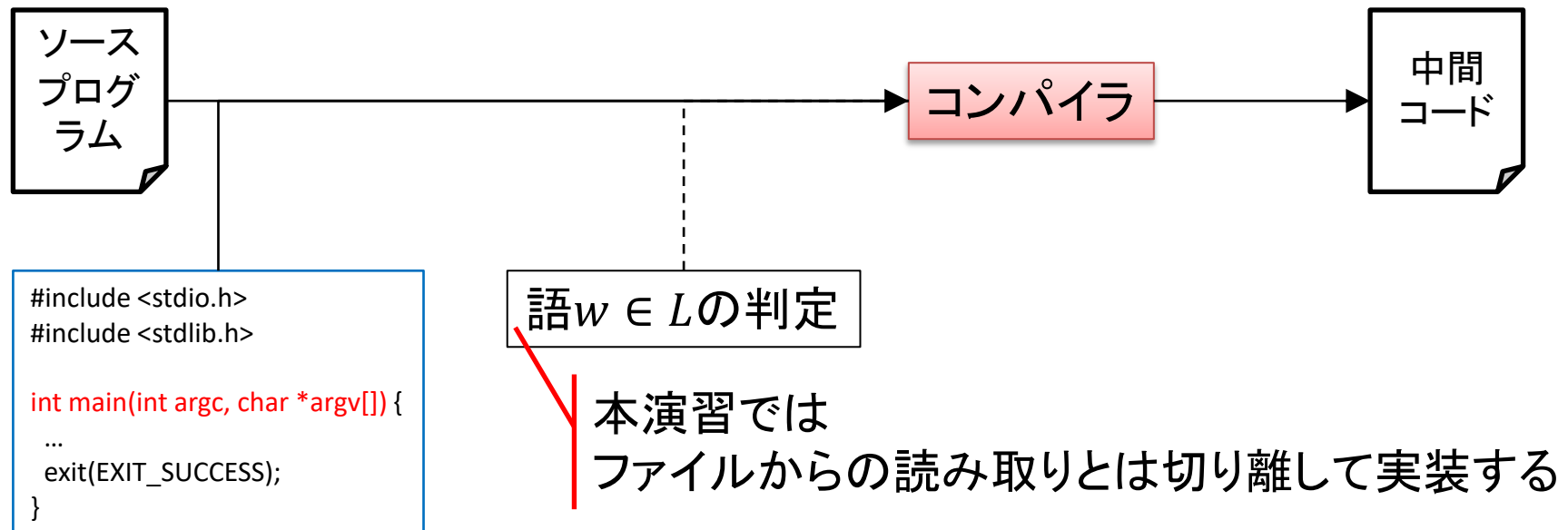
演習2-2: 言語判定機の実装

■ コンパイラは語を認識する必要がある

□ 語 w は Σ^* に含まれるか否か

- 言語 $L \subseteq \Sigma^*$ であるが本演習では Σ^* を考える

□ 本演習では, 語は標準入力から入力する



Step0: 実装の雛型

判定機をStep0から順次拡張して実装する

■ 方針

- 判定に必要な各種機能を関数として実装する



実装の仕方は様々ですが、一部の関数は今後の実装でも使用します。main関数の中に記述せず、演習で指定された関数として実装すること。

Step0: 実装の雛型

■ 判定機を順次実装するプログラムの雛型 "compiler02_step0.c"

□ 機能

- 変数の宣言, main関数のみ
(コンパイルできるが何も表示されないことに注意)

□ グローバル変数

- char Sigma[] // アルファベットの有限集合
- char Str[] // 文字列格納用の配列



プログラミングとしてはスマートではありませんが、グローバル変数を用います

Step1: 記号列の読み込み

- 標準入力から記号列(文字列)を読み込む関数getWordの実装
 - 記号列はグローバル変数Strに格納する
 - Strに格納した記号列を標準出力に表示する
 - "compiler02_step1.c"

関数名	getWord	
引数	void	
戻り値	void	
機能	グローバル変数のchar型配列Str[]に標準入力から文字列を読み込む。ただし、最大文字数は256文字と仮定する。	

Step2: 記号の取得

- 記号列の先頭から記号(1文字)を順に取得する関数nextCharの実装
 - 終端記号'¥0'に注意
 - "compiler02_step2.c"

関数名	nextChar	
引数	void	
戻り値	char	str[]に含まれる次の1文字
機能	<p>グローバル変数として宣言されるchar型配列str[]に対して、次の1文字を返す.</p> <ul style="list-style-type: none">• n回目に関数が呼ばれたとき, str[]のn番目の文字. ただし, nがstr[]に格納されている文字より大きい場合, '¥0'を返す.	

Step3: 文字の判定

- 1文字がアルファベットであるか調べる関数 `isAlphabet`の実装
 - アルファベット Σ は定義されているとする
 - ・ グローバル変数 `char Sigma[]`
 - 戻り値はbool型
 - "compiler02_step3.c"

関数名	isAlphabet	
引数	char c	アルファベットであるか調べる文字
戻り値	bool	
機能	文字cが, グローバル変数として宣言されているSigmaの要素である場合true, そうでない場合falseを返す.	

Step4: 仕上げ

- Step1～3を用いて判定機を仕上げる
 - "compiler02_step4.c"
 - コンパイルしてエラーがないことを確認せよ
 - 動作テストを行ない, 文字列の判定が正しく行われることを確認せよ

課題

■ 演習2-1

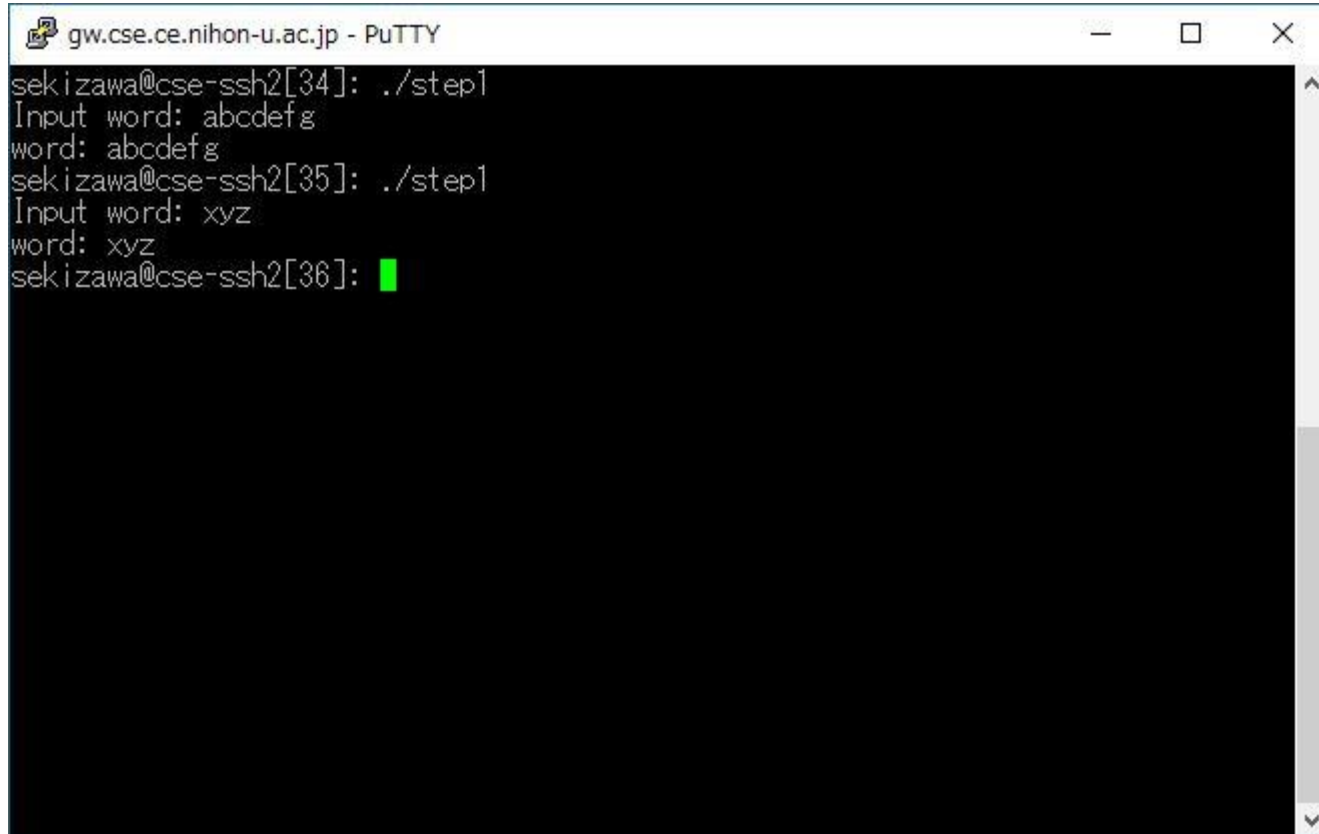
- 演習時間中に解答
- 提出: 演習時間終了時に回収

■ 演習2-2, Step1 ~ Step4

- 課題提出システムに提出

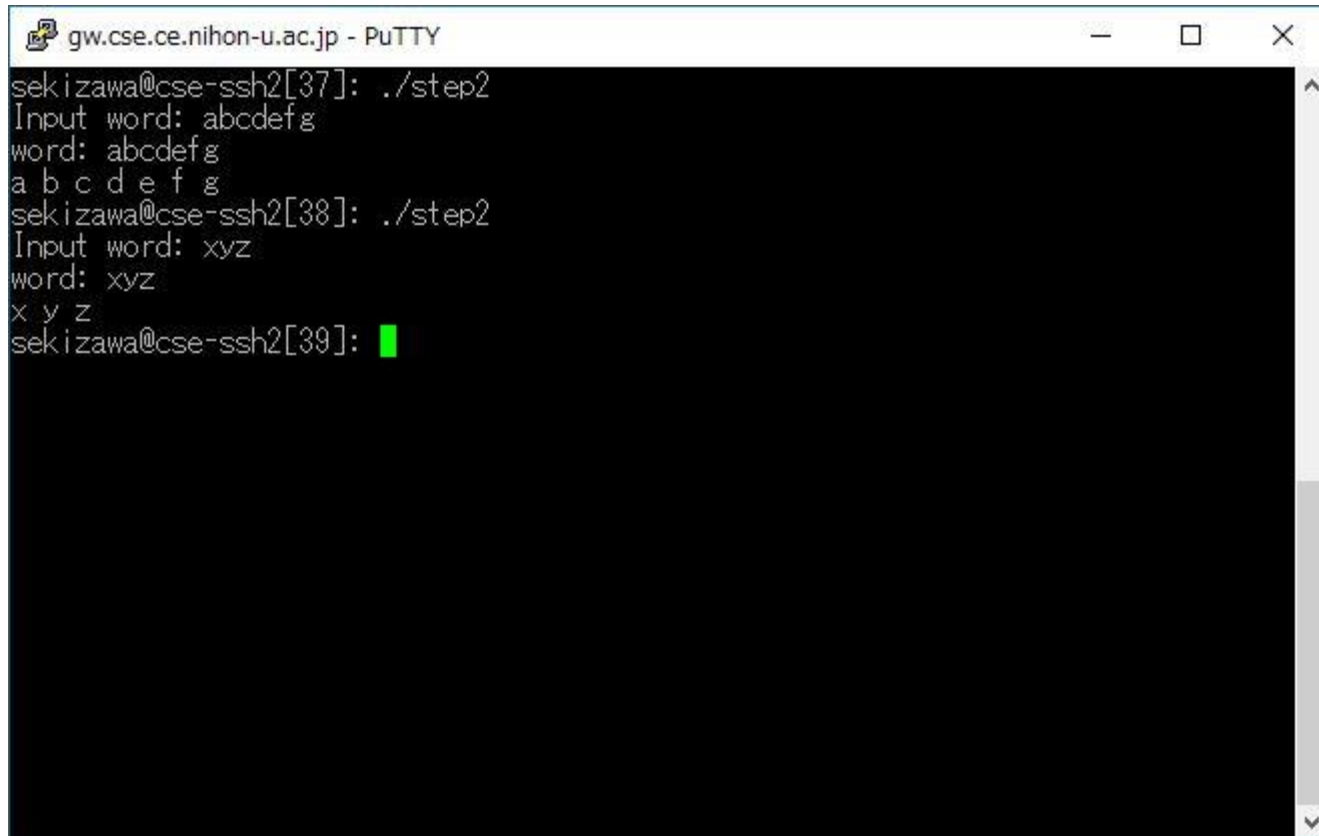
<http://info.ce.nihon-u.ac.jp/compiler/>

Step1: 動作例



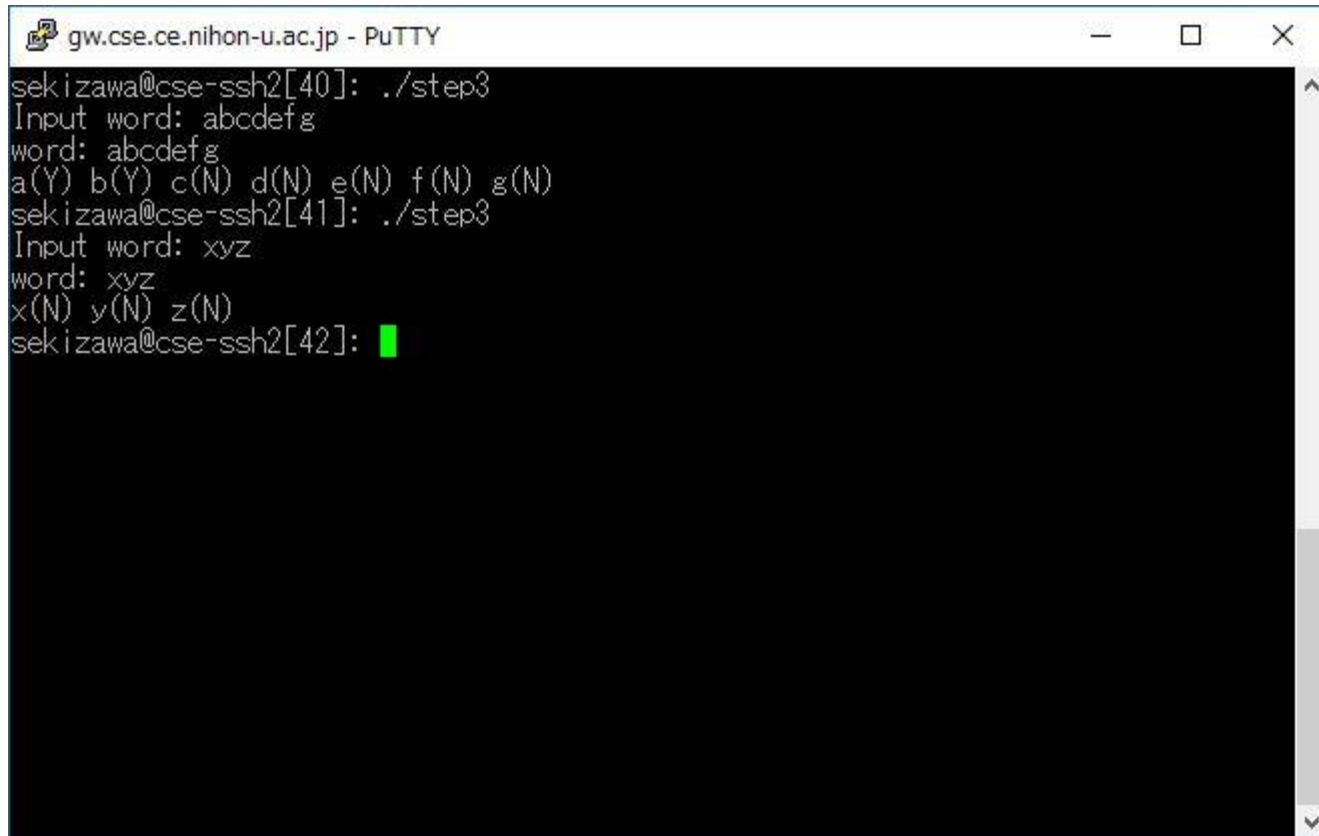
```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh2[34]: ./step1
Input word: abcdefg
word: abcdefg
sekizawa@cse-ssh2[35]: ./step1
Input word: xyz
word: xyz
sekizawa@cse-ssh2[36]: █
```

Step2: 動作例




```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh2[37]: ./step2
Input word: abcdefg
word: abcdefg
a b c d e f g
sekizawa@cse-ssh2[38]: ./step2
Input word: xyz
word: xyz
x y z
sekizawa@cse-ssh2[39]: █
```

Step3: 動作例



```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh2[40]: ./step3
Input word: abcdefg
word: abcdefg
a(Y) b(Y) c(N) d(N) e(N) f(N) g(N)
sekizawa@cse-ssh2[41]: ./step3
Input word: xyz
word: xyz
x(N) y(N) z(N)
sekizawa@cse-ssh2[42]: █
```

Step4: 動作例(1)



```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh2[43]: ./step4
Input word: a
word: a
a is a word of Sigma
sekizawa@cse-ssh2[44]: ./step4
Input word: b
word: b
b is a word of Sigma
sekizawa@cse-ssh2[45]: ./step4
Input word: aa
word: aa
aa is a word of Sigma
sekizawa@cse-ssh2[46]: ./step4
Input word: abababab
word: abababab
abababab is a word of Sigma
sekizawa@cse-ssh2[47]:
```

Step4: 動作例(2)

```
gw.cse.ce.nihon-u.ac.jp - PuTTY
sekizawa@cse-ssh2[48]: ./step4
Input word: aababbba
word: aababbba
aababbba is a word of Sigma
sekizawa@cse-ssh2[49]: ./step4
Input word: abc
word: abc
abc is NOT a word of Sigma
sekizawa@cse-ssh2[50]: ./step4
Input word: xyz
word: xyz
xyz is NOT a word of Sigma
sekizawa@cse-ssh2[51]: ./step4
Input word: AB
word: AB
AB is NOT a word of Sigma
sekizawa@cse-ssh2[52]: ./step4
Input word: abcbba
word: abcbba
abcbba is NOT a word of Sigma
sekizawa@cse-ssh2[53]: █
```