

CS第1 テーマ1

テーマ1の目標

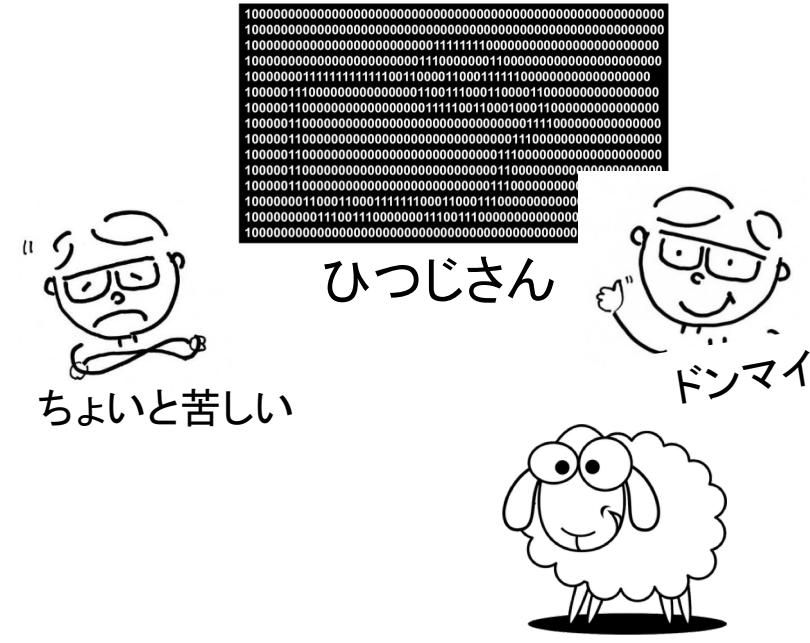
計算の基本要素を知る

演習課題

四則演算でアニメーション

本日の内容

1. はじめに
2. データ = 数 教科書 1.1
3. コンピュータの中では 教科書 2.1
4. Ruby でプログラミング
 - ・ プログラムの基本
 - ・ 分岐, 繰り返し



宿題

1. はじめに

CSのこころ

すべては計算である

- ・ $(1+4) \times 5 =$
- ・ 12 と 16 の最大公約数
- ・ $x^2 + 2xy + y^2$ の因数分解
- ・ 原子炉の設計図を作成する
- ・ 遺伝子を解析する
- ・ 銀行のATMの制御
- ・ 木の成長
- ・ 脳の形成

ただし、コンピュータに
載せるには ...

→
必須

対象をデータとして表すこと
処理を基本演算の組合せで表すこと

2. データは数である

教科書 1.1

データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

確かめてみよう（例で考える）

・数 18, -5, 3.25, 1/3

・文字 a ← 01100001 (=97), b ← 01100010 (=98)

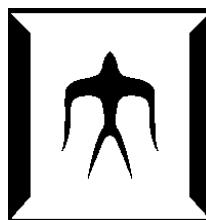
・画像

・音

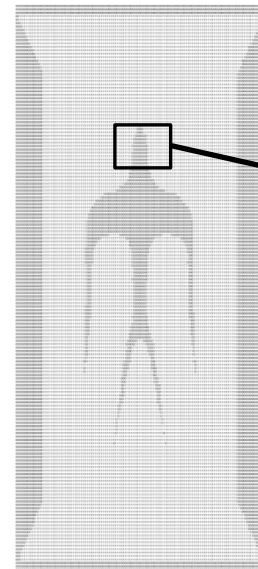
・映像

....

・味, におい？？



→



1111111111111111
1111111111111111
1111111111111111
1111110011111111
1111100001111111
1111100001111111
111100000001111111
111100000001111111

0と1の列

ASCII という符号法

2. データは数である

教科書 1.1

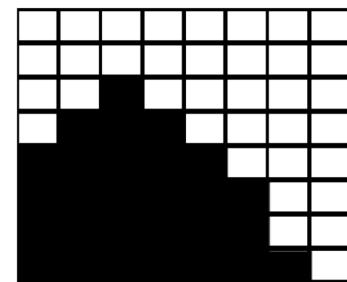
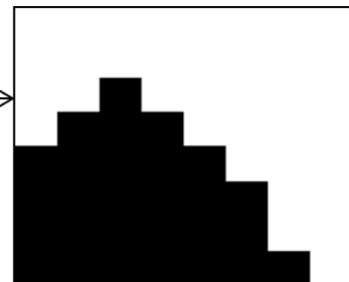
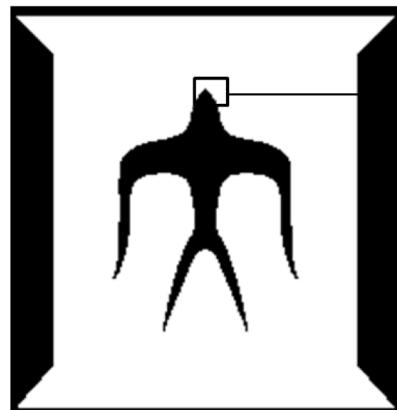
データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

確かめてみよう（例で考える）

- ・ 数
- ・ 文字
- ・ 画像



0と1の列

0と1の列

ツバメの一部を拡大してみた図（左）と
さらに方眼メモリを当てはめた図（右）

デジタル化とは
方眼紙をあてることなり

2. データは数である

教科書 1.1

データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが二進列

確かめてみよう（例で考える）

・数 18, -5, 3.25, 1/3

0と1の列

・文字 $a \leftarrow 01100001$ (=97), $b \leftarrow 01100010$ (=98)

ASCII という符号法

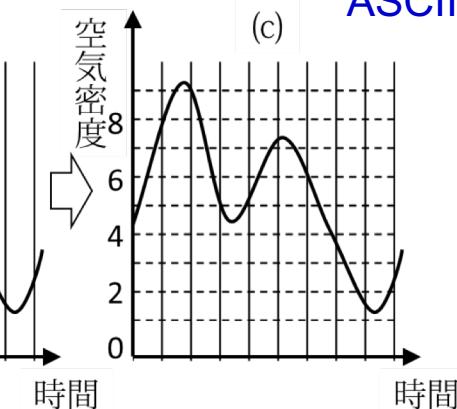
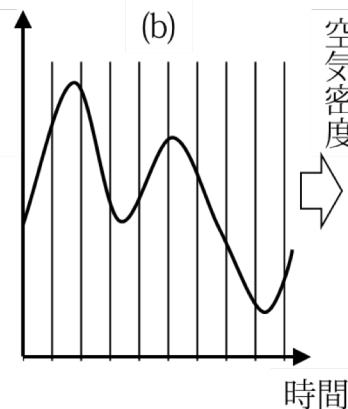
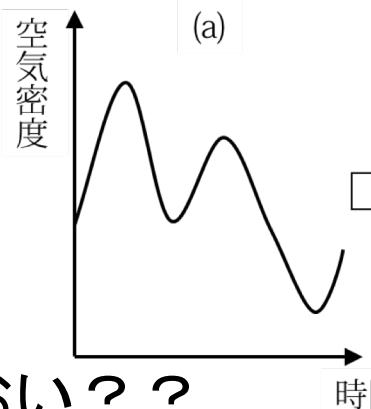
・画像

・音

・映像

....

・味, におい? ?



2. データは数である

教科書 1.1

データ = 計算の対象

どこかで聞いたね

コンピュータの中ではすべてが ~~二進列~~

確かめてみよう（例で考える）

- ・ 数 18, -5, 3.25, 1/3
- ・ 文字 $a \leftarrow 01100001$ ($=97$)

- ・ 画像
- ・ 音
- ・ 映像

....

話を簡単に
するため

0と1の列

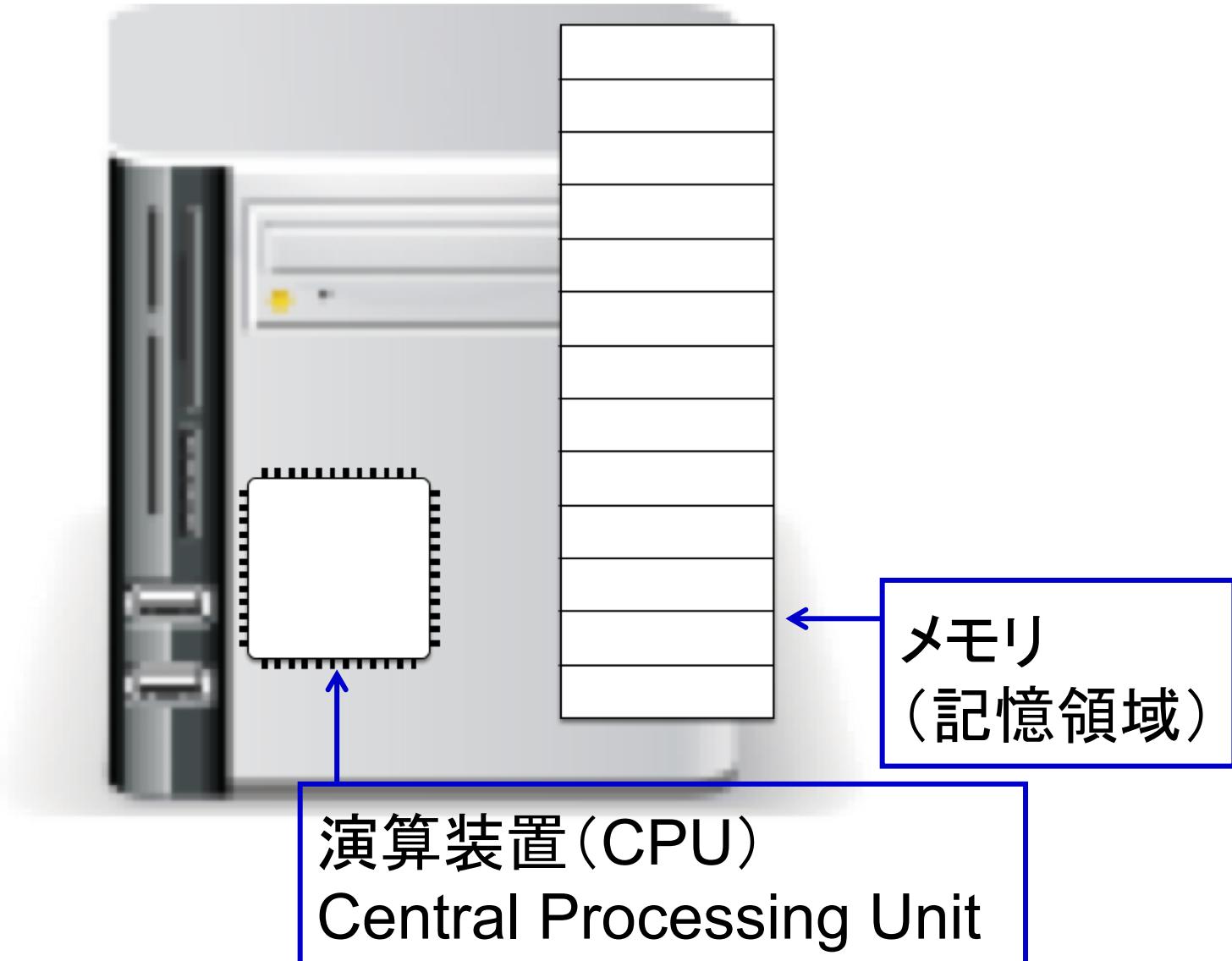
自然数

0, 1, 2, 3, ...

3. コンピュータの中では

教科書 2.1

コンピュータの基本 = 演算装置とメモリ



3. コンピュータの中では

教科書 2.1

メモリ = データ(数)を
格納する箱



演算装置

1234	[0]
78	[1]
0	[2]
123	[3]
9542	[4]
:	
0	[9998]
0	[9999]

メモリ

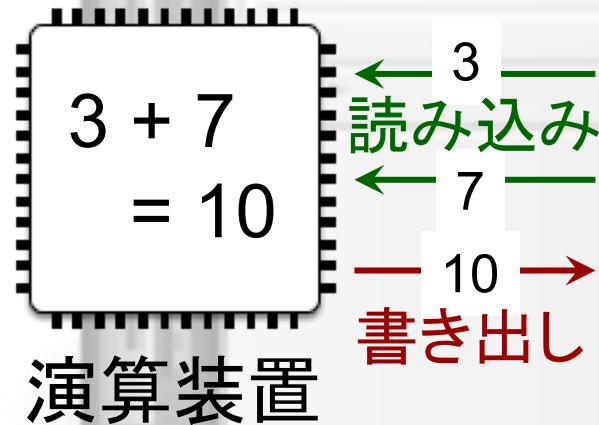
データ

番地

3. コンピュータの中では

教科書 2.1

演算装置 = 基本演算を行った装置



プログラム

```
lw $x, M(100)  
lw $y, M(101)  
add $z, $x, $y  
sw $z, M(102)
```

lw \$x, M(100)
lw \$y, M(101)
add \$z, \$x, \$y
sw \$z, M(102)
3
7
10
:
0
0

メモリ

誰が指示するの？

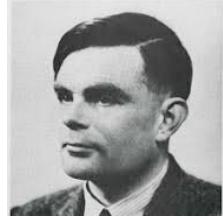
プログラムも
メモリに格納
される

3. コンピュータの中では

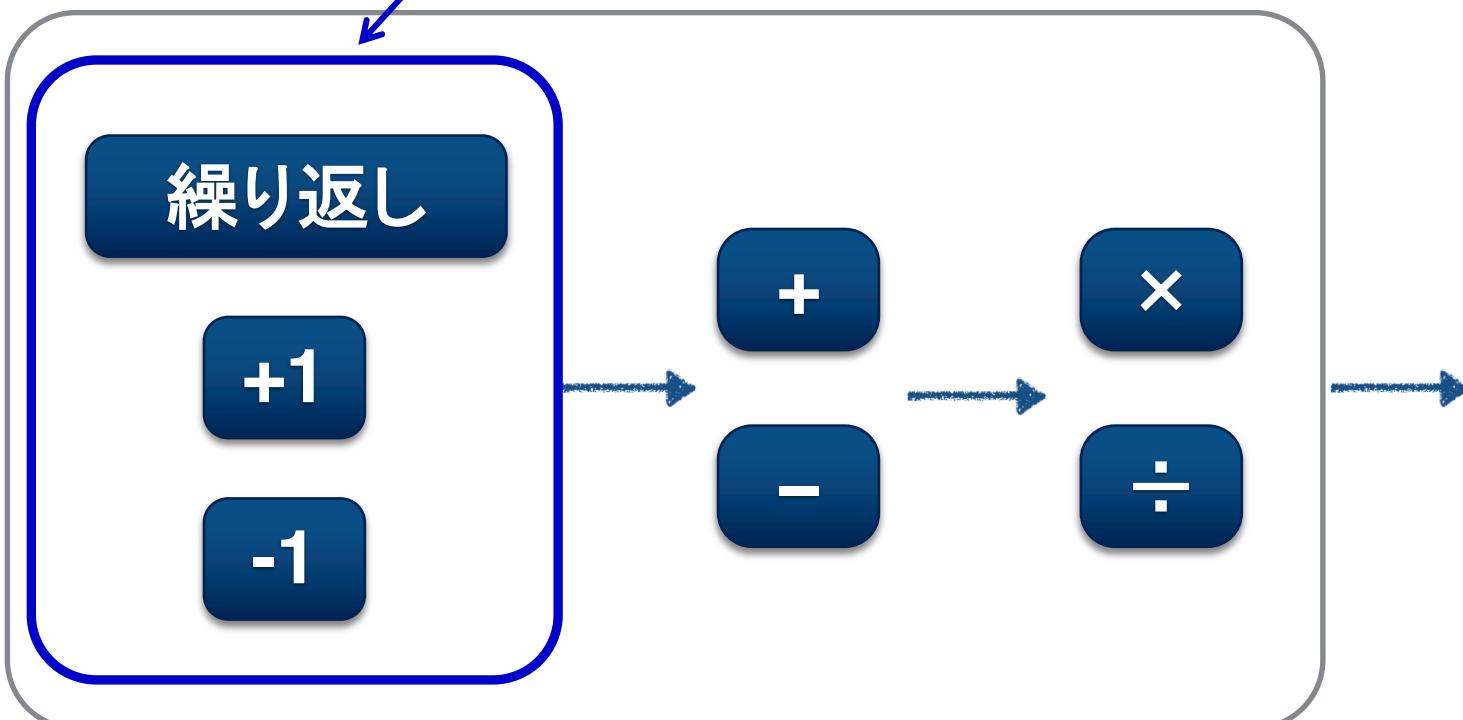
教科書 2.1

演算装置 = 基本演算を行う装置
とは？

計算の原子に
相当する物は何？



チューリング



四則演算

囲碁, 将棋

3. コンピュータの中では

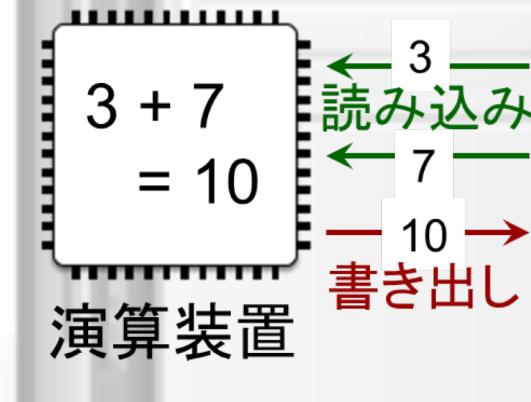
教科書 2.1

入出力 = 外とのやり取り

プログラム

入力 ⇒

⇒ 出力



コンピュータ内部
でのやり取り

```
lw $x, M(100)  
lw $y, M(101)  
add $z, $x, $y  
sw $z, M(102)
```

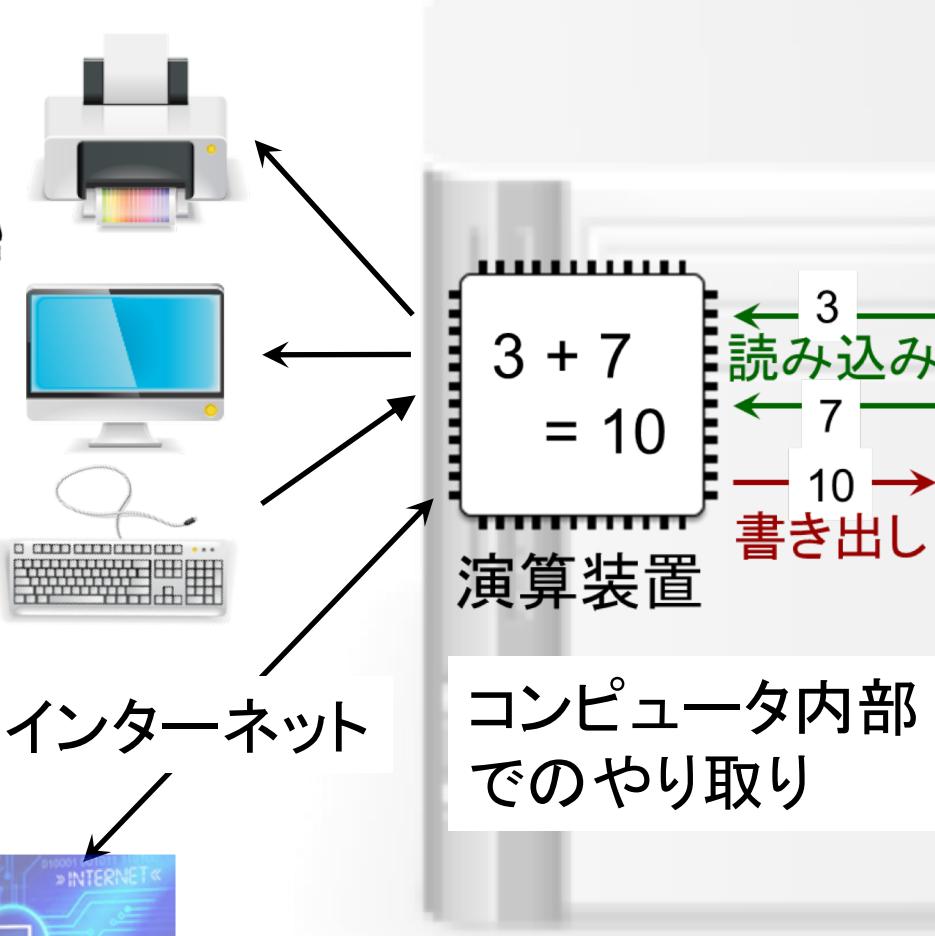
3	[20]
7	[21]
10	[22]
:	
0	[9998]
0	[9999]

メモリ

3. コンピュータの中では

教科書 2.1

入出力 = 外とのやり取り



プログラム

```
lw $x, M(100)  
lw $y, M(101)  
add $z, $x, $y  
sw $z, M(102)
```

3

[20]

7

[21]

10

[22]

:

0

[9998]

0

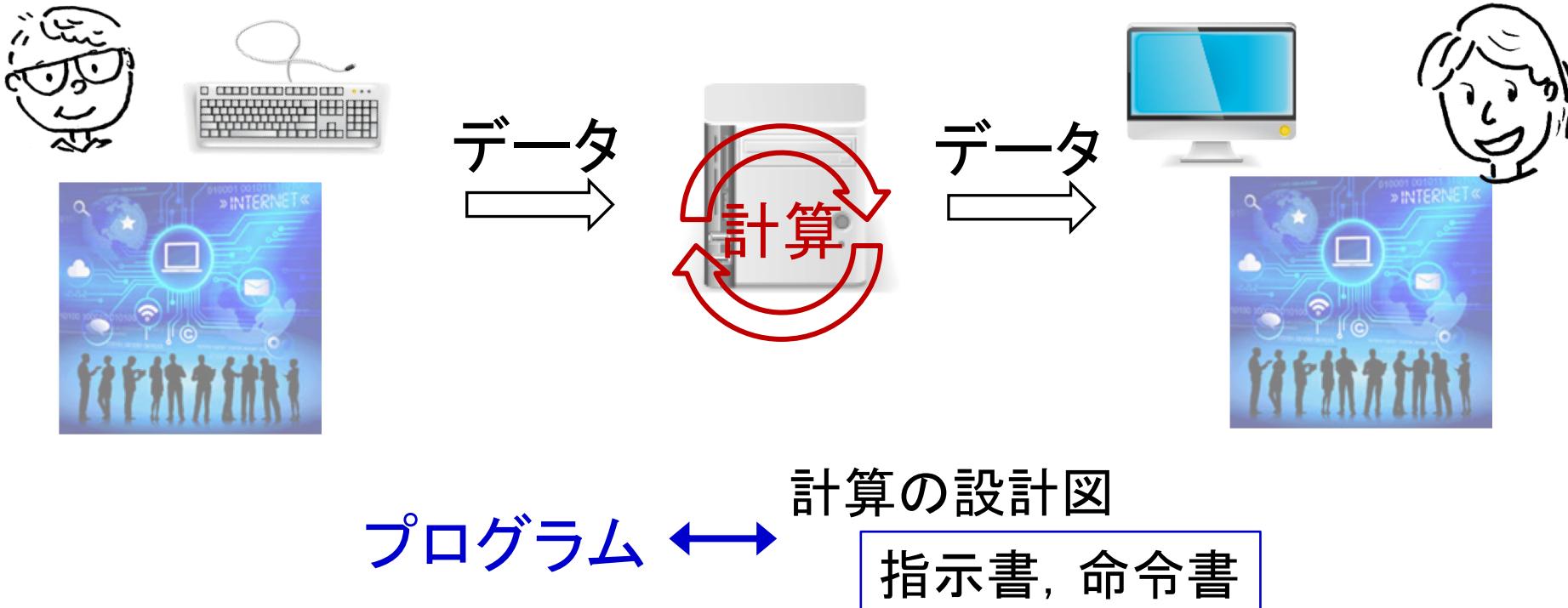
[9999]

メモリ

ここまでまとめ

データ = 自然数

計算 = 超基本要素は ±1 と繰り返し



Ruby でプログラミング !!

※プログラミング=プログラムを作ること

プログラム \leftrightarrow 計算の手順書

プログラムの例：平方根を求めるプログラム

sq.rb

```
n = gets().to_i  
a = 1; a2 = a * a  
while a2 <= n  
  a = a + 1  
  a2 = a * a  
end  
puts(a - 1)
```

データの入力

実際の計算の部分

データの出力

実行例

```
$ ruby sq.rb  
26  
5  
$
```

Ruby \leftrightarrow

プログラミング言語(プログラムを書く言葉)
の1つ。まつもと氏が開発した言語

4. Ruby での書き方

プログラム(指示書)一般に共通するルール

- ・ プログラムはプログラム名のファイルに格納
- ・ プログラムは命令の列
- ・ 上から下に順に実行(繰り返し, 条件分岐は例外)

sq.rb (例)

```
n = gets()
a = 1; a2 = a * a
while a2 <= n
    a = a + 1
    a2 = a * a
end
puts(a - 1)
```

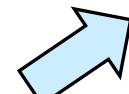
プログラム(指示書)の基本要素

- ・ **変数** \longleftrightarrow データの格納場所
- ・ 基本命令は
 - (1) 変数への代入文
 - (2) 繰り返し文
 - (3) 条件分岐文

add_8_3.rb

```
a = 8
b = 3
wa = a
while b > 0
    wa = wa + 1
    b = b - 1
end
puts(wa)
```

Ruby での書き方のルールを例を参考に説明しましょう



8 + 3 を求める計算をするプログラム

4. Ruby での書き方(その1)

代入文 \leftrightarrow 変数に値を格納せよ, という命令

【文の構造】

変数名 = 計算式

例) $abc = 3 * (4 - de) + 6 / x * y$

abc

2

de

2

x

3

y

例) $b = b - 1$

3

b

add_8_3.rb

a = 8

b = 3

wa = a

while b > 0

 wa = wa + 1

 b = b - 1

end

puts(wa)

8 + 3 を求める計算
をするプログラム

4. Ruby での書き方(その1, 補足)

【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

- この講義では**整数変数**(整数格納用の変数)のみとする。
(最初に整数を入れると整数変数と認識される。※1)
- 整数変数が入った割り算の答えは整数(切り捨て値)となる。

※1 Ruby 特有の方便。本格プログラムではお勧めできない。

※2 Ruby 以外の言語では ** は使えない場合も多い。

4. Ruby での書き方(その2)

繰り返し文 **while** 文

↔ 指定の条件が成立する間繰り返せ
という命令

【文の構造】

while 条件

繰り返される命令列
(複数の文もOK)

end

例)

add_8_3.rb

a = 8
b = 3
wa = a

while b > 0
 wa = wa + 1
 b = b - 1
end
puts(wa)

8 + 3 を求める計算
をするプログラム

4. Ruby での書き方(例)

掛け算を±1と繰り返しのみで実現する

mult.rb

```
x = 入力データ  
y = 入力データ  
seki = 0  
while y > 0  
    seki = seki + x  
    y = y - 1  
end  
puts(seki)
```

add.rb (参考)

```
a = 入力データ  
b = 入力データ  
wa = a  
while b > 0  
    wa = wa + 1  
    b = b - 1  
end  
puts(wa)
```

±1以外も使ってるよ

4. Ruby での書き方(例)

掛け算を±1と繰り返しのみで実現する

mult.rb

```
x = 入力データ  
y = 入力データ  
seki = 0  
while y > 0  
    seki = seki + x  
    y = y - 1  
end  
puts(seki)
```

```
a = seki  
b = x  
wa = a  
while b > 0  
    ...  
    b = b - 1  
end  
seki = wa
```

```
x = 入力データ  
y = 入力データ  
seki = 0  
while y > 0  
    wa = seki; b = x;  
    while b > 0  
        wa = wa + 1  
        b = b - 1  
    end  
    seki = wa  
    y = y - 1  
end  
puts(seki)
```

宿題: Ruby でプログラミング

次のプログラムを紙に書いてくること

- (1) 引き算を±1と繰り返しのみで計算する
- (2) 割り算を加減算と繰り返しのみで計算する

提出方法

- ・ 紙に書いてくる
※配布した用紙を使ってもよい、説明は不要
- ・ 次回の授業開始まで
※授業開始の **20分後**に受付締め切ります
- ・ 採点法: 提出することが大切！

4. Ruby での書き方(その3)

条件分岐文 if 文

↔ 条件に応じて分岐する
(計算の流れを変える)命令

【文の構造】

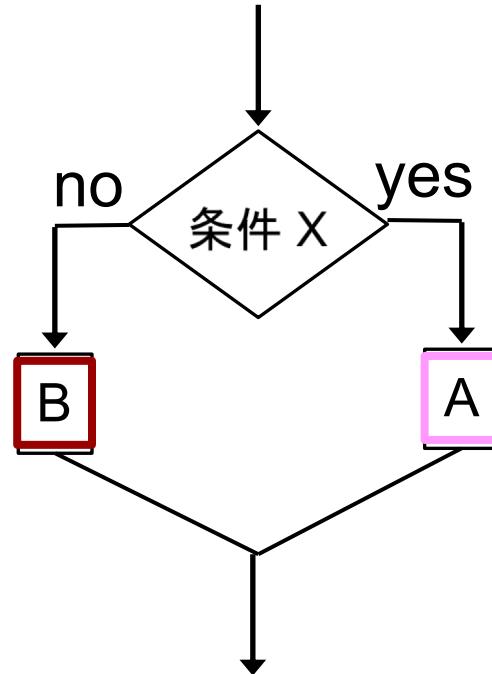
if 条件 X

 命令列 A

else

 命令列 B

end



abs.rb

```
# abs.rb
x = 入力された数
if x >= 0
    puts( x )
else
    puts( 0 - x )
end
```

絶対値を求める計算

※ 究極的には if 文は不要 (while 文で代用できる).

4. Ruby での書き方(その3)

条件分岐文 if 文

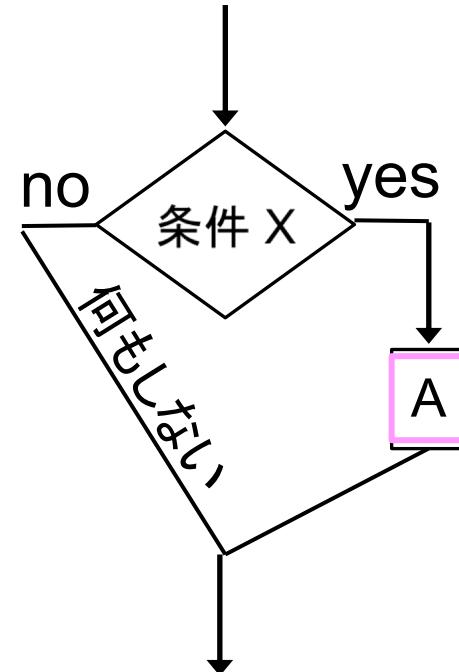
↔ 条件に応じて分岐する
(計算の流れを変える)命令

【文の構造】(変形版)

if 条件 X

命令列 A

end



abs.rb

```
# abs.rb
x = 入力された数
if x < 0
    x = -x
end
puts( x )
```

絶対値を求める計算

4. Ruby での書き方(その3; 補足)

条件式 \leftrightarrow 条件を指定する式. while 文, if 文などで使う

例) while $a \geq x^{**} 2$

if $b \% 2 \neq 0$

【関係演算子】

関係	使用例	意味
\geq	$x \geq y$	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
\neq	$x \neq y$	x は y は等しくない
$<$	$x < y$	x は y より小さい
\leq	$x \leq y$	x は y より小さいかまたは等しい

※ 究極的には $>$ だけに限ってもよい(つまり、他を言い換えられる).

まとめ: Ruby (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
\geq	$x \geq y$	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
\equiv	$x \equiv y$	x は y と等しい
\neq	$x \neq y$	x は y は等しくない
$<$	$x < y$	x は y より小さい
\leq	$x \leq y$	x は y より小さいかまたは等しい

まとめ: Ruby (2)

【論理演算子】

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