情報数学
第1回　WHILEプログラム

萩野　達也
hagino@sfc.keio.ac.jp

授業概要

- コンピュータプログラムは入力をもらって出力を出すという意味では数学の関数とみなすことができるが，この授業ではプログラムに対応する関数の性質につて取り上げる
- まず，プログラムにより計算可能な関数を理解するために，プログラムの3つのモデル，帰納的関数，チューリング機械，ラムダ計算を紹介する．これら3つのモデルは同等である．
- 次に，ラムダ計算およびプログラムのモデルを理解するために，完全半順序について取り上げる．
- 最後に，プログラムのデータ型を理解するために，関数を抽象的に取り扱う圏論について紹介する．

授業予定

1. Whileプログラム
2. 原始帰納的関数
3. 帰納的関数
4. チューリング機械
5. チューリング機械と計算可能性
6. ラムダ計算
7. ラムダ計算と計算可能性
8. 完全半順序
9. 完全半順序とデータ型
10. 連続関数
11. 表示的意味論
12. 圏論入門
13. 極限と随伴関手
14. 圏論とデータ型
15. 期末試験

計算とはなにか？

- 計算 = コンピュータができること
- 自然数に関する計算のみを取り扱う.
  - $N = \{0, 1, 2, 3, 4, 5, 6, 7, \cdots\}$
- コンピュータは自然数に関する四則演算（足し算，引き算，掛け算，割り算）を行うことができる.
  - 引き算の場合に負になるときは0 とする.
    - 例）$3 - 5 = 0$
  - 割り算の時は，小数点以下は切り捨てる.
    - 例）$5 \div 2 = 2$
- コンピュータにできること.
  - 四則演算の結果を変数に覚えておく（代入文）.
  - 変数の値を参照しながら四則演算を行う.
  - 手順に従って処理を順番に行う.
  - 変数の値を調べて，手順を変更する（条件文）.

計算手順

- 計算とは
  - 与えられたいくつかの自然数を変数に入れておき，
  - 手順にしたがって演算を行ない，
  - 最後に，ある変数の値を計算の結果とする．

- 数学的には
  - 計算 = コンピュータができること
  - コンピュータは数字を計算する関数とみなすことができる．
  - コンピュータはどのような関数を計算することができるのか？
  - 計算可能な関数とは？

- アルゴリズム（計算手順）= 計算順序を表したもの
  - アルゴリズムはフローチャートとして表すことができる．

最大公約数の求め方

- 2つの自然数の最大公約数を求める
  - 最も大きな公約数(両方の約数)
  - 2つの数を割り切る最大の数
  - 自然数 m と n の最大公約数を $\gcd(m,n)$ とする.

- 例: 315と231の最大公約数
  - 315の約数
    - 1, 3, 5, 7, 9, 15, 21, 35, 45, 63, 105, 315
  - 231の約数
    - 1, 3, 7, 11, 21, 33, 77, 231
  - 315と231の公約数
    - 1, 3, 7, 21
  - 315と231の最大公約数
    - 21

ユークリッドの互除法

- 最古のアルゴリズム
  - ユークリッド：紀元前330年頃～紀元前275年頃
  - ユークリッド原論で有名，その中に出てくる

- 自然数 n と m の最大公約数を求めるユークリッドの互除法

1. n を m で割った余り r を求める.
   - $n = q \times m + r$
   - $\gcd(n, m)$ と $\gcd(m, r)$ は等しい.
2. n, m を m, r で置き換える.
3. これを n が m で割り切れるまで繰り返す.
4. 割り切れて，余り r が 0 になる.
   - $\gcd(n, 0) = n$

ユークリッドの互除法を使った最大公約数の計算

- 例: $\gcd(315,231)$
  - $\gcd(315,231)$
    - $315 \div 231 = 1 \cdots 84$
  - $\gcd(315,231) = \gcd(231,84)$
    - $231 \div 84 = 2 \cdots 63$
  - $\gcd(231,84) = \gcd(84,63)$
    - $84 \div 63 = 1 \cdots 21$
  - $\gcd(84,63) = \gcd(63,21)$
    - $63 \div 21 = 3 \cdots 0$
  - $\gcd(63,21) = \gcd(21,0)$
  - $\gcd(21,0) = 21$

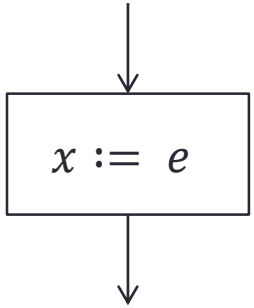
ユークリッドの互除法

1. n を m で割った余り r を求める.
   - $\gcd(n,m) = \gcd(m,r)$
2. n, m を m, r で置き換える.
3. これを n が m で割り切れるまで繰り返す.
4. 割り切れて r が 0 になる.
   - $\gcd(n,0) = n$

$$\frac{231}{315} = \frac{231 \div 21}{315 \div 21} = \frac{11}{15}$$

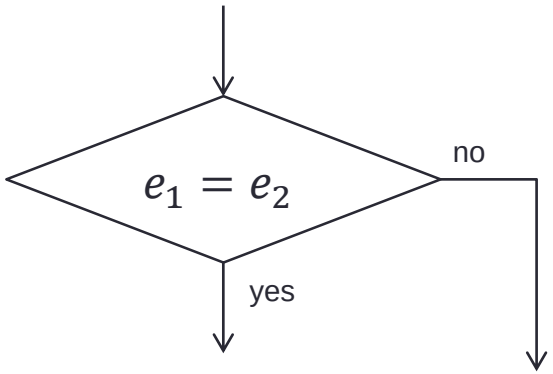
フローチャート

- 代入



e は変数と自然数を含む四則演算の式

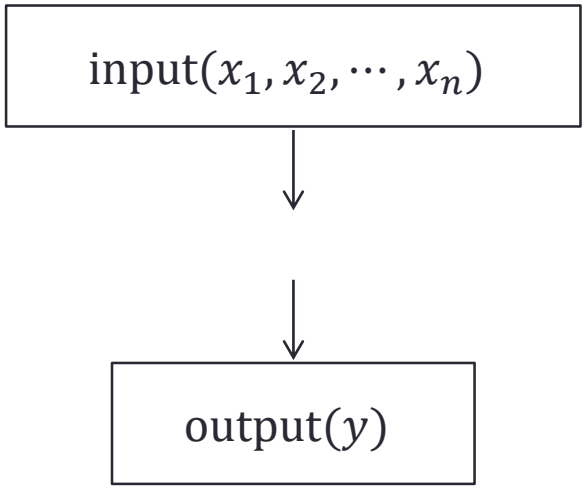
- 条件分岐



e_1 と e_2 は変数と自然数を含む四則演算の式

入出力

- 入力

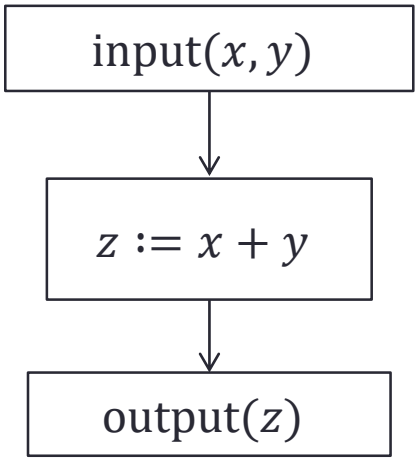


- 出力

- フローチャートプログラム
  - 入力から始まり，代入文と条件分岐を線で結び，出力で終わる．
  - 入力で与えられた変数に対する計算結果を出力とする．

$$f : \underbrace{N \times N \times \cdots \times N}_{\text{入力}} \rightarrow \underbrace{N}_{\text{出力}}$$

簡単なプログラム

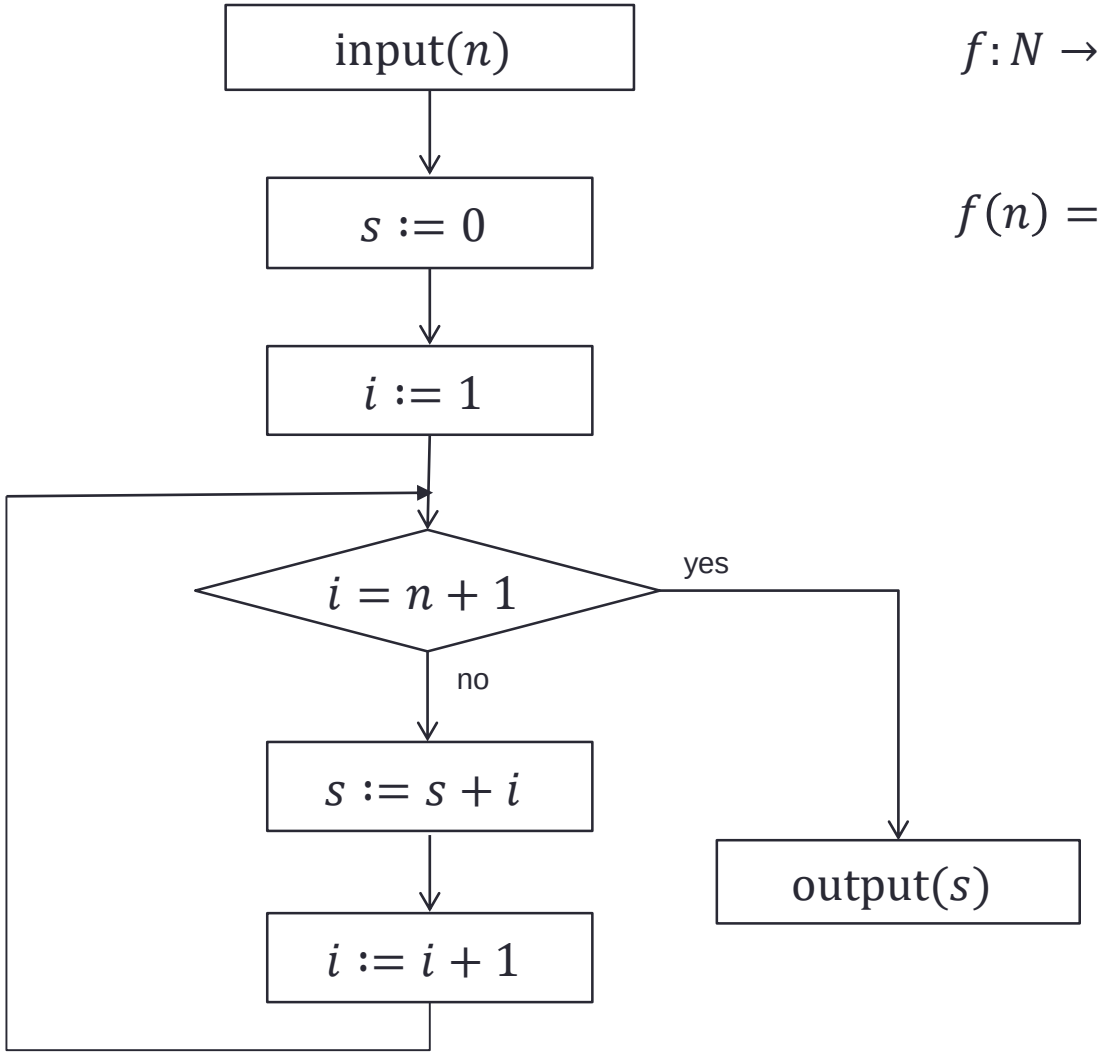


$f : N \times N \to N$

input　output

$f(x, y) = x + y$

1+2+…+n を計算するプログラム



$$f: N \to N$$

$$f(n) = \sum_{i=1}^{n} i$$

ユークリッドの互除法

ユークリッドの互除法を
フローチャートで表しなさい

Whileプログラム

- プログラミング言語
  - 2次元的グラフであるフローチャートをプログラムとしてコンピュータに与えるのは難しい.
  - 1次元的な言語として表したい.

- Whileプログラム
  - $\mathrm{input}(x_1, x_2, \cdots, x_n)$
  - $\mathrm{output}(y)$
  - $x := e$
  - $\{P_1; P_2; \cdots; P_n\}$
  - if $(e_1 = e_2)$ then P else Q
  - while $(e_1 = e_2)$ P

Whileプログラムの例

- 1+2+…+n を計算するプログラム

```
input(n);
s := 0;
i := 1;
while (i <= n) {
  s := s + i;
  i := i + 1
}
output(s);
```

Whileプログラムの例

- ユークリッドの互除法をwhileプログラムとして書きなさい.

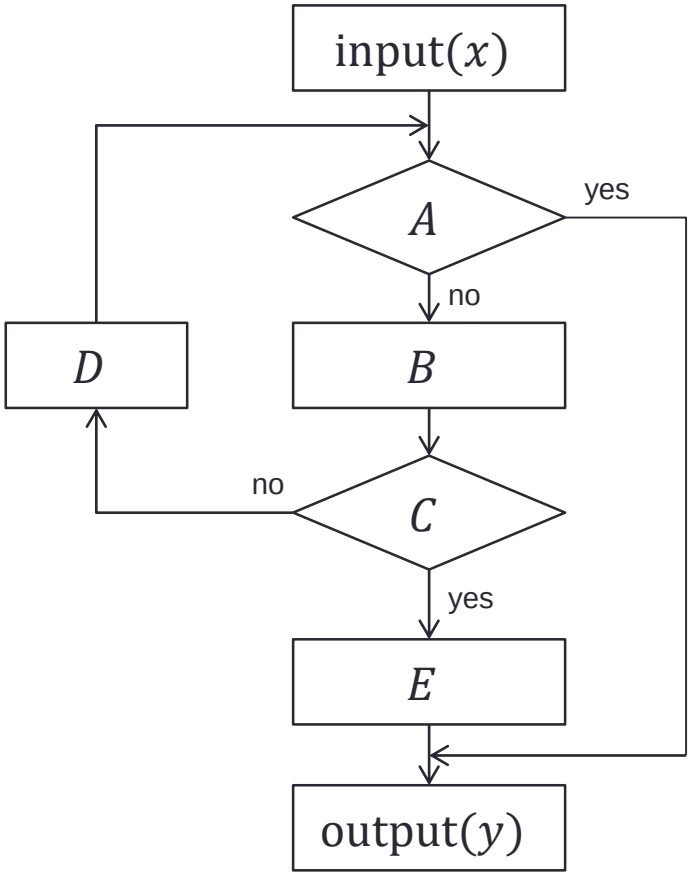
```
input(n,m);




output(n);
```

フローチャートとwhileプログラム

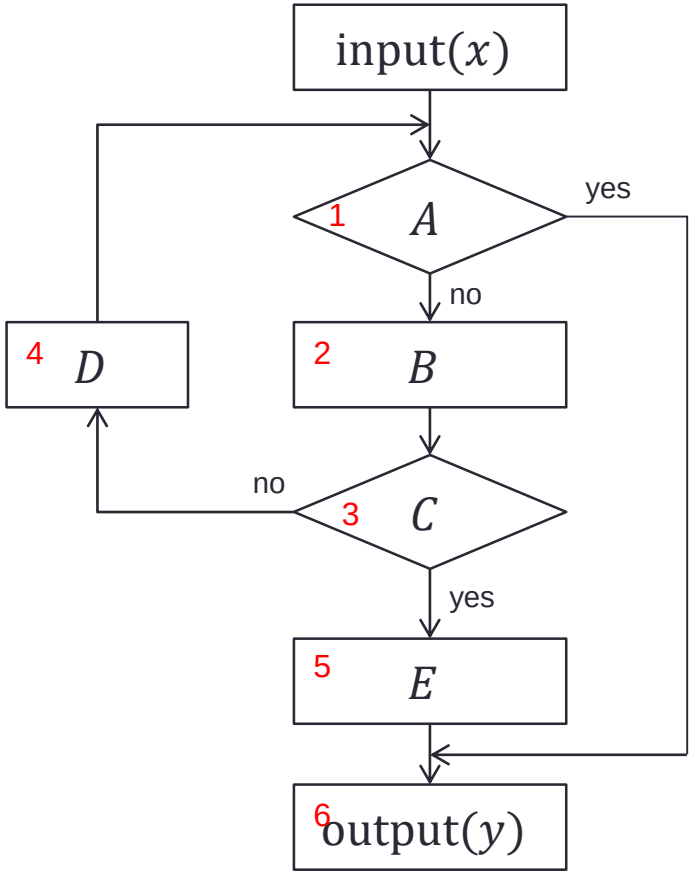
- **定理:**
  - 任意のwhileプログラムはフローチャートで表すことができる.
  - 任意のフローチャートはwhileプログラムで表すことができる.

- **証明:**
  - 任意のwhileプログラムをフローチャートで表すことができることはほとんど自明
  - 逆
    - フローチャートの代入文，条件分岐に箱番号をつける.
    - 次に行なう箱番号を管理する変数を導入する.
    - 代入文，条件分岐の後で次の箱番号を与えるようにする，
    - whileプログラムは，箱番号を管理するプログラムとすれば良い.

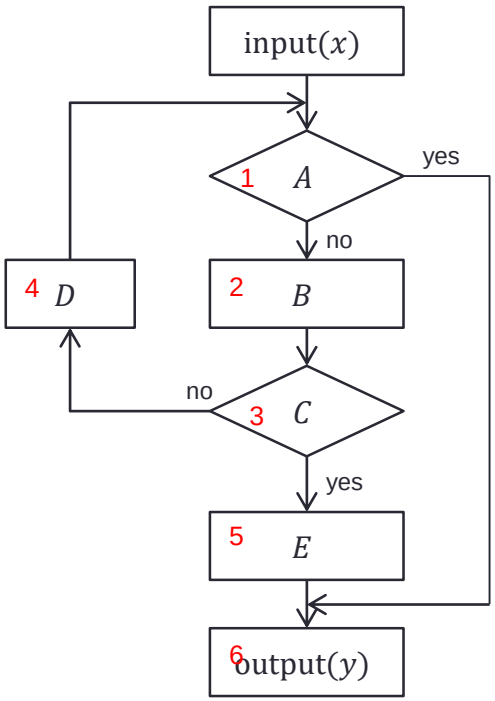
変換例



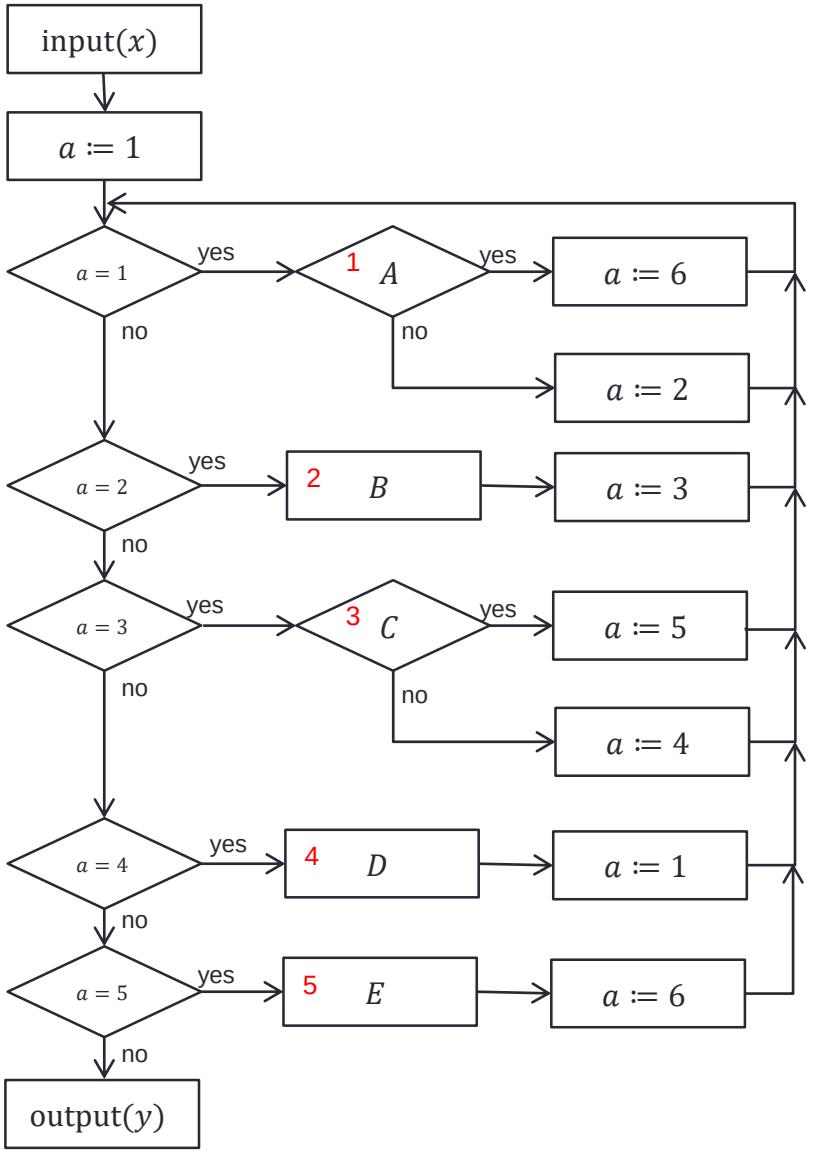
箱に
番号を
付ける



変換例



箱番号を
管理する変数
aを導入



変換例

- whileプログラムとして書き表す

```
input(x);



 a:=1;
 while (a-5=0) {
   if (a=1) then { if (A) then a:=6 else a:=2 }
   else if (a=2) then { B; a:=3 }
   else if (a=3) then { if (C) then a:=5 else a:=4 }
   else if (a=4) then { D; a:=1 }
   else if (a=5) then { E; a:=6 }
 }




output(y);
```

系

- **系:**
  - 任意のwhileプログラムはwhile文が高々一つだけのプログラムに書き換えることができる.

- **証明:**
  - あたえられたwhileプログラムをフローチャートにする.
  - フローチャートを前の定理にしたがいwhileプログラムに変換する.

まとめ

- 計算 = コンピュータが行う計算
- 計算可能関数 = コンピュータが計算する数学的な関数
- 計算可能性 = 数学的な関数をコンピュータが計算できるか
  - すべての数学的な関数をコンピュータが計算できるわけではない.
  - 計算することのできない関数が存在する.